(2)

NAVAL POSTGRADUATE SCHOOL
Monterey, California

THESIS

A HYPERMEDIA APPROACH TO THE DESIGN OF AN INTELLIGENT TUTORING SYSTEM

by

Elizabeth M. McGinn

September 1991

Thesis Advisor Yuh-jeng Lee

REPORT DOCUMENTATION PAGE

1a. REPORT SECURITY CLASSIFICATION Unclassified	1b. RESTRICTIVE MARKINGS
2a. SECURITY CLASSIFICATION AUTHORITY	3. DISTRIBUTION/AVAILABILITY OF REPORT Approved for public release; distribution is unlimited.
2b. DECLASSIFICATION/DOWNGRADING SCHEDULE	
4. PERFORMING ORGANIZATION REPORT NUMBER(S)	5. MONITORING ORGANIZATION REPORT NUMBER(S)
6a. NAME OF PERFORMING ORGANIZATION Naval Postgraduate School 6b. OFFICE SYMBOL (*If Applicable*) 37	7a. NAME OF MONITORING ORGANIZATION Naval Postgraduate School
6c. ADDRESS (*city, state, and ZIP code*) Monterey, CA 93943-5000	7b. ADDRESS (*city, state, and ZIP code*) Monterey, CA 93943-5000
8a. NAME OF FUNDING/SPONSORING ORGANIZATION 6b. OFFICE SYMBOL (*If Applicable*)	9. PROCUREMENT INSTRUMENT IDENTIFICATION NUMBER
8c. ADDRESS (*city, state, and ZIP code*)	10. SOURCE OF FUNDING NUMBERS PROGRAM ELEMENT NO. / PROJECT NO. / TASK NO. / WORK UNIT ACCESSION NO.

11. TITLE (*Include Security Classification*)
A HYPERMEDIA APPROACH TO THE DESIGN OF AN INTELLIGENT TUTORING SYSTEM

12. PERSONAL AUTHOR(S)
MCGINN, Elizabeth Marie

13a. TYPE OF REPORT	13b. TIME COVERED	14. DATE OF REPORT (*year, month,day*)	15. PAGE COUNT
Master's Thesis	FROM TO	1991 September	126

16. SUPPLEMENTARY NOTATION
The views expressed in this thesis are those of the author and do not reflect the official policy or position of the Department of Defense or the U.S. Government.

17. COSATI CODES			18. SUBJECT TERMS (*continue on reverse if necessary and identify by block number*)
FIELD	GROUP	SUBGROUP	Intelligent Tutoring Systems, Hypermedia, Computer-aided Instruction, Multimedia, Cognitive Theory

19. ABSTRACT (*Continue on reverse if necessary and identify by block number*)

A hypermedia approach to designing intelligent tutoring systems (ITS) is presented. Animated "page-turners" limited to one knowledge domain represent the majority of ITSs that have been developed in the past several years. They have failed to improve the value or effectiveness of training delivery systems. A need exists to expand present microworld, "page-turner" computer-based instruction applications into complete curricula capable of providing an array of teaching strategies and learning activities for students--in short, a comprehensive instructional environment. A tutorial module designed with hypermedia capabilities, supported by multimedia devices, could generate a more comprehensive set of knowledge-based explanations for students as well as provide a richer learning environment for them to explore. In addition to providing an interactive learning environment for students, a hypermedia-based tutorial module can be an aid to instructors in curricula design. The applicability of hypermedia concepts was reviewed and applied with a prototype Linguist Workstation being developed by the Basic Military Language Course (BMLC) Project at the Defense Language Institute (DLI), Presidio of Monterey.

20. DISTRIBUTION/AVAILABILITY OF ABSTRACT	21. ABSTRACT SECURITY CLASSIFICATION
[X] UNCLASSIFIED/UNLIMITED [ ] SAME AS RPT. [ ] DTIC USERS	Unclassified
22a. NAME OF RESPONSIBLE INDIVIDUAL Yuh-jeng Lee	22b. TELEPHONE (Include Area Code) (408) 646-2361 22c. OFFICE SYMBOL CSLe

DD FORM 1473, 84 MAR — 83 APR edition may be used until exhausted — All other editions are obsolete — SECURITY CLASSIFICATION OF THIS PAGE Unclassified

Approved for public release; distribution is unlimited.

A Hypermedia Approach to the Design of an

Intelligent Tutoring System

by

Elizabeth Marie McGinn
Lieutenant, United States Navy
B.S., United States Naval Academy, 1985
Ed. M., Boston University, 1989

Submitted in partial fulfillment of the requirements
for the degree of

MASTER OF SCIENCE IN COMPUTER SCIENCE

from the

NAVAL POSTGRADUATE SCHOOL
September 1991

Author: _____
Elizabeth Marie McGinn

Approved By: _____
Yuh-jeng Lee, Thesis Advisor

_____
David R. Pratt, Second Reader

_____
Robert B. McGhee, Chairman,
Department of Computer Science

ABSTRACT

A hypermedia approach to designing intelligent tutoring systems (ITS) is presented. Animated "page-turners" limited to one knowledge domain represent the majority of ITSs that have been developed in the past several years. They have failed to improve the value or effectiveness of training delivery systems. A need exists to expand present microworld, "page-turner" computer-based instruction applications into complete curricula capable of providing an array of teaching strategies and learning activities for students--in short, a comprehensive instructional environment. A tutorial module designed with hypermedia capabilities, supported by multimedia devices, could generate a more comprehensive set of knowledge-based explanations for students as well as provide a richer learning environment for them to explore. In addition to providing an interactive learning environment for students, a hypermedia-based tutorial module can be an aid to instructors in curricula design. The applicability of hypermedia concepts was reviewed and applied with a prototype Linguist Workstation being developed by the Basic Military Language Course (BMLC) Project at the Defense Language Institute (DLI), Presidio of Monterey.

TABLE OF CONTENTS

I. INTRODUCTION 1

A. MEDIA: HYPER-, MULTI-, AND INTERACTIVE 1

B. INTEGRATION OF HYPERMEDIA AND EDUCATION 5

C. OBJECTIVES 8

II. AUTOMATED LEARNING ENVIRONMENTS: NEW PERSPECTIVES IN DESIGN 10

A. THE STATE OF COMPUTER-AIDED INSTRUCTION 10

B. A HYPERMEDIA APPROACH 13

1. Expert module 14

2. Tutorial module 15

3. Student-diagnostic module 16

4. Interface module 17

5. Instructional Design Aids 18

III. COGNITIVE THEORY, EFFECTIVE INSTRUCTION AND HYPERMEDIA 20

A. CONSTRUCTIVIST COGNITION 20

B. COMMUNICATION KNOWLEDGE 21

1. Traditional person-computer interfaces 21

2. Traditional CAI interfaces 21

3. ITS 22

C. INDIVIDUALIZED INSTRUCTION 22

1. Sponge method 23

2. Cognitive apprenticeship method 23

3. Artist and Exploration Method 24

4. Research method 26

5. Argument method 26

D. INTERACTIVITY 27

1. Interruptability 27
2. Graceful Degradation 27
3. Limited Look-ahead 28
4. No default 28
5. Impression of an infinite database 28
E. CONNECTING THE DOTS: HYPERMEDIA LINKS CONTENT, LEARNING CONTEXT AND LEARNER 29
F. PRESCRIPTION FOR IMPLEMENTATION 30
IV. MULTIMEDIA/AI INTEGRATION AND COMPUTER-ASSISTED LANGUAGE LEARNING 33
A. INTERACTIVE MULTIMEDIA 33
1. Brown University's InterMedia: OOP + Hypermedia 33
2. Perseus 39
B. COMPUTER-ASSISTED LANGUAGE LEARNING: PASSIVE PAGE-TURNERS VS. INTERACTIVE CONTEXTS 41
1. Memory 44
2. Comprehension 44
3. Production 45
4. Testing 45
V. THE LINGUIST WORKSTATION 47
A. FRAMEWORK FOR DEVELOPMENT 47
B. DESIGN RATIONALE AND DESCRIPTION 48
C. HARDWARE AND SOFTWARE 51
D. DEVELOPING COURSEWARE IN PHASES 55
E. LINGUIST WORKSTATION INTERFACE 56
F. THE VIDEO LESSONS: PROTOTYPES IN GERMAN & KOREAN 62
G. VIDEO LESSON TEMPLATE: VIEWING FEATURES 66
VI. CONCLUSIONS 76
A. ACCOMPLISHMENTS, CURRENT LIMITATIONS AND FUTURE RESEARCH 76

B. A CASE FOR HYPERMEDIA LEARNING ENVIRONMENTS..... 77

APPENDIX A 80

APPENDIX B 104

REFERENCES 114

INITIAL DISTRIBUTION LIST..... 119

I. INTRODUCTION

Most of the prior research in the field of intelligent tutoring systems (ITS) has focused on applications that are limited to specific knowledge domains [Sl82, Ha88, Ps88, Ra90]. A limited knowledge base restricts the ability of a tutoring system to present students with a variety of teaching strategies. The ability to adapt to the diverse learning styles of individual students and offer diverse explanations which cross many knowledge domains is a trait characteristic of "good" human educators. These "good teaching principles and behaviors" have proven extremely difficult to codify and program into computer-based instruction systems. Formulating answers to all possible questions, for even one specific body of knowledge (e.g., basic geometry proofs, steam plant emergencies, etc.) quickly becomes a combinatorially explosive problem [Br82, Ha88, Ps88, Ra90, Sl82, Wf91].

Animated "page-turners" limited to one knowledge domain represent the majority of ITSs that have been developed in the past several years [Br82, Ps91, Ra90, Sc91]. They have failed to improve the value or effectiveness of training delivery systems [Ge87, Sc91]. A need exists to expand present microworld, "page-turner" computer-based instruction applications into complete curricula capable of providing an array of teaching strategies and learning activities for students--in short, a comprehensive instructional environment.

A. MEDIA: HYPER-, MULTI-, AND INTERACTIVE

The growth of hypermedia concepts in software development holds promise for removing some of the difficulties encountered in the past when building the

"intelligence" of a human educator into computer-based instruction systems. However, before investigating the instructional design alternatives that hypermedia might offer, it is essential to understand the nature of hypermedia, hypertext, and multimedia. Rather than strictly reserving definitions for terms that are used interchangeably in both industry and academia, this section describes general characteristics of "hypermedia-based" systems.

In a word, hypertext and hypermedia can be described as *nonsequential.* Unlike sequential programs, text in books, or video in films, there is no pre-determined order in which hypertext or hypermedia information must be viewed. A hypertext-based system is designed to provide alternatives for users to explore rather than a single stream of information: its major characteristic is that information is displayed to the user as determined by the user, dynamically--that is, as the user interacts with the system [Bi90, Bo91, Er91, It90, Ka90, Me86, Ni90, Pe91, Ps91].

Hyper*text* conventionally refers only to text-based systems, whereas hyper*media* denotes a variety of media formats: graphics, text, sound, and video [Bi90, Bo91, Er91, In90, Ka90, Ni90, Pe91, St91]. Hypertext can be thought of as both the precursor to and a subset of hypermedia. Hypermedia has been called "multimedia hypertext," and is considered a "natural technique for supporting multimedia interfaces since it is based on the interlinking of nodes that may contain different media." [Ni90] Interactive multimedia also refers to the presentation and communication of diverse modes of information to a user. As in hypermedia systems, dialogue between user and computer system (i.e., the user interface) determines the retrieval and display of information. Hence, the term "interactive" was added to some categories of multimedia applications to

distinguish them from more passive forms of multimedia presentations (e.g., a slide show with photos, graphics, and text). The naming convention used in this thesis is as follows: hypermedia describes the theoretical, internal methodology for data representation, while interactive multimedia will be used to refer to actual applications and/or implementation of the methodologies.

Structurally, hypermedia can be represented as a network of nodes and links (Figure 1.1). A node represents some bit of data--a word, a graphic, a sound, a video segment. A link connects one node to another in the same sense that a "pointer" data structure points from one field of information to another. The links between objects *presented* to the user (visually or aurally--e.g., windows, text, or sounds) and objects *represented* in a database formulate the key concept of hypermedia. As defined in *Intelligent Assistant Systems* [By91],

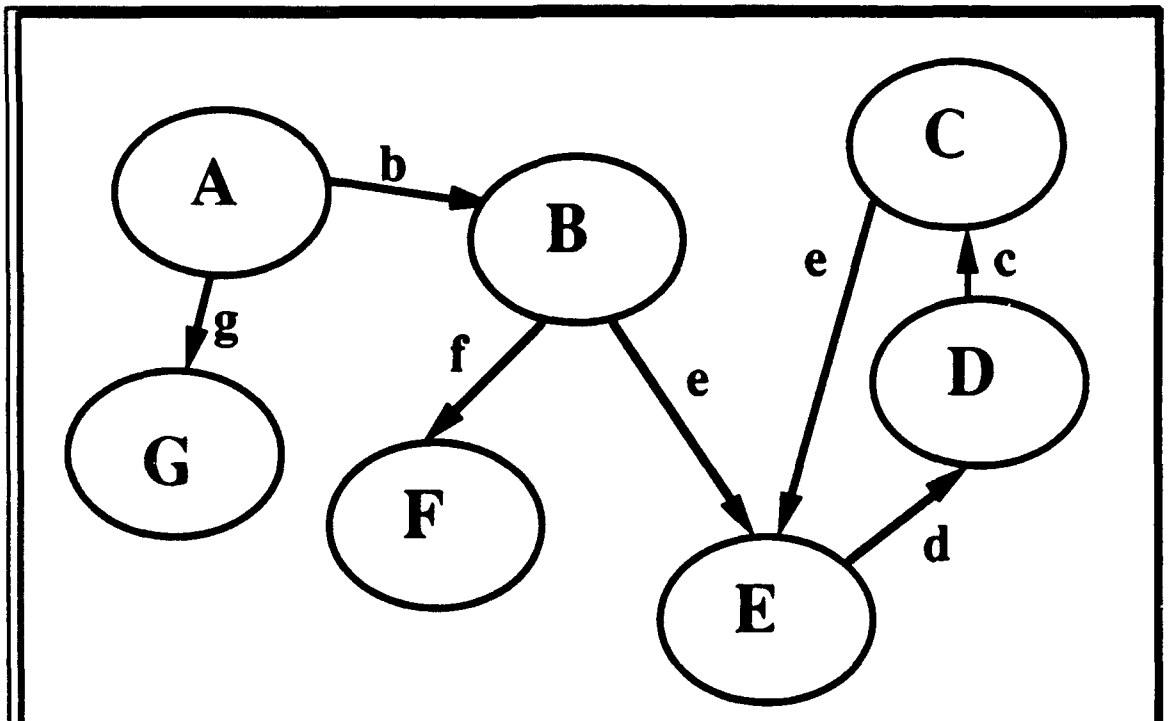

Figure 1.1 Hypermedia structure having seven nodes and seven links.

A HyperText (sic) system may be described as a system including a database which is a network of textual or graphical nodes, and windows on the screen corresponding one-to-one with nodes in the database. A HyperText database is a directed graph.

Figure 1.2 expands Figure 1.1 to illustrate this mapping. Application features such as "hot words" within bodies of text, "hotspots" within graphics objects, buttons, and icons are all examples of hypermedia display nodes [By91, Ni90]. A user activates a link between nodes by "clicking on," touching, or selecting a display node in some way that is compatible with the user interface.

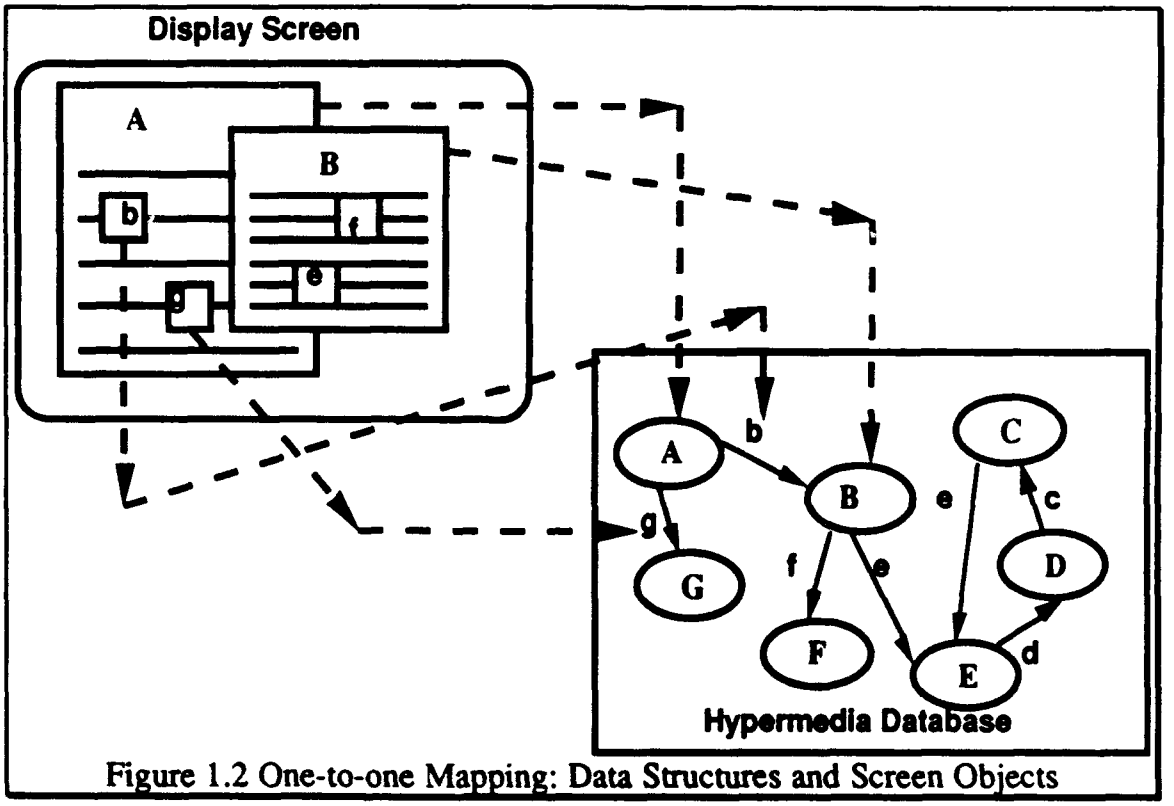

Figure 1.2 One-to-one Mapping: Data Structures and Screen Objects

Because hypermedia systems provide the user with conceptual connectivity (the links) between database objects and screen objects, they offer more than

conventional database management systems (DBMS), which only link information nodes within a database. Hypermedia systems also differ from windowing systems, which have no underlying database. A multi-window editor may allow movement by users among several bits of information for comparison on a screen. However, users must invoke the extra windows themselves. In hypermedia, links represented by hotspots or icons are represented internally as well, so that the "computer finds the information for the user." [Ni90] The terms "browsing" and "navigating" are used to describe how a user progresses throughout this network of information. They emphasize that the user actively determines the order the nodes are visited instead of merely following a pre-determined program sequence [Bi90, Bm91, Bo91, Er91, Ka90, Me86, Pe91, Ps91, St91, Wo91].

B. INTEGRATION OF HYPERMEDIA AND EDUCATION

As described, hypermedia enables data retrieval via referential links defined by a programmer/user in a dynamic, highly interactive way. Ultimately, various knowledge representation schemes could be created by the power of nonlinear association of information. Multiple knowledge representation schemes enable flexibility in representation and presentation of teaching points. Such flexibility of design is essential for instructional developers, who must sequence the acquisition of knowledge, as well as for students, who must assimilate and associate the information presented with meaningful, coherent conceptual links. Further, the prevailing assumption in education theory is that the learning experience is enhanced when more than one medium (or "mode") of communication is used to illustrate a concept [Af90, Bg85, Ba91, Bo91, Er91, It90, Pe91]. Thus, to be an effective teaching tool, an ITS should be capable of

representing information/knowledge in multiple formats. However, in the past, flexibility in presentation and represention of knowledge has been difficult to design and to program [Ps88, Sw91, Wf91]. Consequently, very few ITSs have incorporated these features in their overall design [Ps88, Ra90].

Recent advances in hardware technology can extend the application of hypermedia to knowledge representation design. Audio boards and video cards, along with peripheral devices such as audio recorders, CD-ROMs, and videodiscs allow various types of media to be incorporated in both microcomputer and distributed systems' applications. Integration of these components enables the creation of multimedia learning environments. Research conclusions support application of this idea: "Hypertext provides a systematic approach to structuring and delivering knowledge organized in complex networks and presented in multi-media (sic) formats." [Ps91] A tutorial module designed with hypermedia capabilities, supported by multimedia devices, could generate a more comprehensive set of knowledge-based explanations for students as well as provide a richer learning environment for them to explore.

In addition to providing an interactive learning environment for students, a hypermedia-based tutorial module can be an aid to instructors in curricula design. Object-oriented programming (OOP) concepts, sophisticated graphics interfaces, and hypermedia linking facilities can be integrated to help instructors design effective, individualized learning environments. Authoring applications ("authorware") now available commercially can facilitate the instructional design process and provide low-cost prototype development systems [Bi90, Bo91, Cr91, Er91, Ga91, In90, Ph91, Pe91, St91, Wo91]. When integrated with commercial spreadsheet or database management utilities, multimedia authoring tools can

provide a system that will help instructors monitor student progress more effectively. Ultimately, data from these student profiles will enable instructors to tailor lesson plans according to the learning styles of individual students. For example, "knowledge links completed" by each student can be stored in student profiles and accessed by instructors.

Currently, there exist software tools that can be used to implement such tutoring systems for the military [Af90, Bm91, Cr89, Fa90, Ps91]. The portability of computer software and compatibility of most of today's microcomputer-based systems provide strong justifications for the implementation and standardization of computer-based training (CBT) for all branches of military service. Nowhere is the maintenance of a high level of technical proficiency and expertise more important than the military service: due to high personnel transfer rates, commands regularly lose experienced, technical experts and individuals with valuable corporate knowledge while gaining "green troops" who require indoctrination. Continuous military training and a standard method of instructional delivery are thus high priority tasks; however, loss of key personnel from operational commands to remote training units can have a major impact on mission effectiveness and operational tempo. Implementation of a computer-based military training system will help to facilitate exportation of instruction to operational commands DoD-wide, which will serve both to standardize training and to decrease any training costs associated with travel time and loss of personnel.

C. OBJECTIVES

This thesis examines two primary consequences of previous knowledge representation and acquisition limitations and offers alternative perspectives for the design, development, and delivery of computer-based instruction:

Observed Consequences:	Proposed Solutions:
1) Most of the ITSs that have been produced, both in academic research labs and for commercial use, are "add-on" meta-rule routines designed to generate explanations and manage the knowledge databases of existing expert systems--applications whose expert knowledge is limited to specific domains, and whose response bandwidth and interactive features are neither flexible nor dynamic;	1) Centralize knowledge resources in an interactive, hypermedia-based learning environment by integrating application development software and expert systems technology;
2) the design and development of most courseware currently available for educators outside the research labs is characterized by large amounts of programming for small amounts of lesson material.	2) develop instructional design "toolkits" that provide hypermedia-linking facilities that can be used both in the production of lesson plans and in the evaluation of student progress.

Specifically, this thesis will address the architecture design, and rationale for development of a multilevel, multilingual computer-based learning laboratory at the Defense Language Institute (DLI), Presidio of Monterey.

Current research data on language learning theory suggests that properly implemented multimedia technologies can significantly facilitate foreign language learning [Cr89, Bm91, Ps91, Cr, in press]. However, many of the programs

now available for foreign language instruction appear to be computerized versions of the traditional grammar-translation based textbooks [Dk91, Ps91]. The easy integration of sound and video into current software packages has only recently become possible; hence, the development of comprehensive language tutorials has not progressed very far.

These resources are available now at DLI: a sound theoretical foundation and comprehensive strategy must be developed to focus the effort. Prototype modules in Hebrew and Russian, written in Apple's *Hypercard*, are currently being designed and tested in the Education Technology Branch language labs at DLI [Fa90]. Multimedia tutorials and instructional design modules in German and Korean using *Multimedia Extensions to* Microsoft *Windows* are being developed as the automated component of the Basic Military Language Course (BMLC) Project at DLI, a project established to implement DLI's long-term plan to modernize foreign language instructional formats [Bm91].

II. AUTOMATED LEARNING ENVIRONMENTS: NEW PERSPECTIVES IN DESIGN

A. THE STATE OF COMPUTER-AIDED INSTRUCTION

The use of computers in education is known by many different names: Computer-based Training (CBT), Computer-Aided Instruction (CAI), Intelligent Computer-Aided Instruction (ICAI), Intelligent Tutoring Systems (ITS). Although these names are meant to distinguish between different types of instructional systems, the primary purposes of any of them are fundamentally the same: to create an environment conducive to learning, and to help educators design tools to realize such environments.

The first aids to automate instruction were CAI systems, described as electronic "page turners," devoid of any intelligence regarding the sequencing, selection, and presentation of subject matter [Br82]. When expert system shells offered a knowledge representation base for instructional designers, some CAI systems were dubbed "intelligent;" hence, the "I" in ICAI and ITS. Classic artificial intelligence techniques such as production rules and semantic nets were implemented to manage the expert knowledge bases for instruction presentation and sequencing. A generic architecture and design methodology began to emerge [Br82, Sl82, Ps88].

The "foundational anatomy" of an ITS consists of the following components: the expert module, which must contain representations of the domain knowledge; the student diagnostic module, which must track what the student knows based on how he performs and interacts with the system; the interface module, whose

primary importance lies in its transparency to the student/user--so that a student need not learn a computer interface to learn a concept or skill; and the tutorial module, which determines the sequencing and selection of instruction, based on student profiles created by the student module [Bu91]. Figure 2.1 depicts the classic ITS architecture.

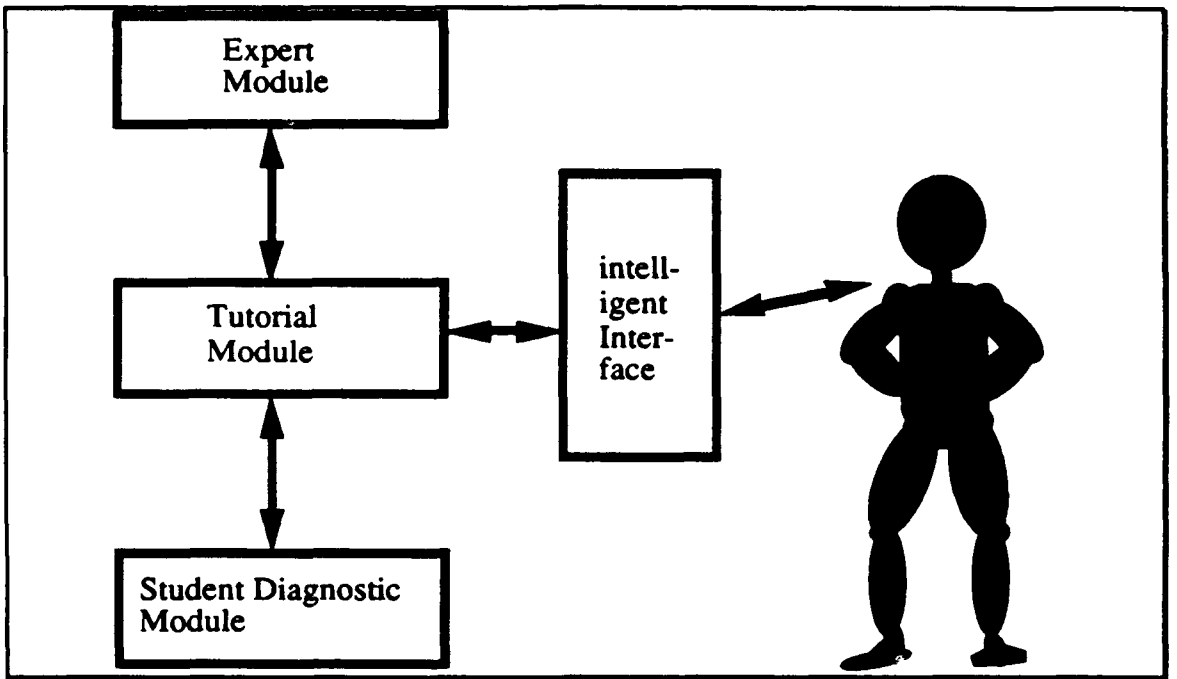

Figure 2.1 Classic ITS Architecture

This classification of components by function led to separate bodies of research for each module: research on how best to represent the expert, or domain knowledge; how best to record what students know and how they proceed through the learning process; how best to sequence the presentation of expert knowledge and when to test; and how best to present an interface for a student [Br82, Ps88, Ha88, Sw91]. Researchers then focused on the integration of these components. The major assumption driving research held that once the optimum design of each distinct module was determined, their interaction could

be orchestrated by an "intelligent" instructional strategist [Bu91, In89, Wo91]. Given that our theories of cognition, learning, and even effective teaching styles are far from complete between humans, the codification of such all-encompassing knowledge into computers now appears a misplaced research goal to make.

Again, the fundamental question to be answered when designing, developing, and delivering CBT (or CAI, or ICAI, or ITSs) is "How do we provide an environment that facilitates learning?" During the Spring Symposium on Knowledge-Based Environments for Teaching and Learning, assessments of existing systems and speculations about second generation ITSs inspired a new direction for ITS research:

> Researchers have moved away from building omniscient tutors capable of detecting all possible errors and misconceptions. Instead, research is now focused on building empathetic partners that choose from among several forms of interaction based on the content of the communication and the needs of the student. [Wf91]

"Good" teachers communicate with their students. Effective instructors interact at levels commensurate with aptitudes and learning styles of their pupils. Dynamic dialogue between student and computer and dynamic interaction among ITS modules has become the focus for a new generation of ITSs [Bu91, Sw91, Wf91]. The current postulate for ITS designers is "that the primary purpose of an ITS is to provide the student with a set of operators that will cause or support the communication of knowledge." [Sw91] To realize this objective in an implementation, the designer must focus on the interface and the tools that drive it: "the designer of intelligent tutors is responsible for building a massive

communication management system that controls and monitors a student's learning environment." [Sw91]

B. A HYPERMEDIA APPROACH

An interactive hypermedia-based system may provide an appropriate testbed for these new approaches to ITS architecture. Research has concluded that we currently lack the capability to encode an expert to control knowledge, or the imparting of knowledge [Bu91, Wf91]. We have not been able to construct explicit rules to handle every possible question, as tutorial anticipatory strategies are not complete. The new perspective centers round a shared plan of communication between computer as learning environment and student as an active participant in his/her knowledge acquisition. Just as industry converges on applying a "user-centered" strategy in personal computer design methodologies, education concentrates on "student-centered" tutoring designs [Af90, Bo91, Bu91, Cr91, It91, Ph91, Sc91]. A student-centered system offers the student control of the navigation through knowledge bases and interaction over a framework of tutorial suggestions.

The "tutorial suggestions" are explicit instructions to control overall lesson sequencing. They contain no specific lesson content; rather, they are templates, or roadmaps that define and determine the conceptual links that a student must follow when navigating through a lesson. In a computer system that offers multimedia peripherals, instructional templates might contain instructions to control the playing of a videodisc, or audio-player. A student might navigate through the tutorial roadmap via menu items or "radio button" choices in a window interface environment [Af90, Sc91, Ka90]. Initially, these knowledge representation links would be programmed explicitly by instructors with

instructional authoring tools (similar in function to commercially available application authoring tools). Ultimately, meta-rules could be defined that generate explanations and determine lesson sequencing based on an analysis of archived student profiles. The "omniscient tutor" might then be a conceivable, approachable goal.

Current trends in research regarding ITS architectures emphasize the interactivity and integration of components whereas previous research focused on the individual functions of each module. A hypermedia system's power lies in its facilities for linking various media in flexible but controlled ways [By91, Ni90, Ps91]. These facilities can be applied to each module individually as well as to an ITS overall--the integration of components is implicit in hypermedia's network approach to system design. When reconsidering the functions of each module as summarized in the previous section, a hypermedia design methodology provides implementation support for each.

1. Expert module

The primary function of this module, also known as the knowledge representation module, is to represent domains of knowledge, both declarative and procedural knowledge (i.e., facts/concepts and the steps a student must follow in order to apply these facts/concepts). Conventional knowledge representation techniques include graphic simulations, qualitative models, (sequential) text and hypertext. A representation can be defined as "a structure of objects, relationships, and operations together with a mapping that places this structure in some explicit relationship." [Ps91] A knowledge representation network demands the development of structures that represent causal connections explicitly and and can decompose systems into simpler causal structures in the

knowledge hierarchy where functional relationships are more apparent. Hotspot links provide a natural conceptual framework for such representation and presentation. Access to these knowledge structures must be integrated with all of the other ITS modules as well. Hypermedia offers the most general and powerful solution to these requirements.

2. Tutorial module

The functions of this module include how best to select and sequence instruction, adapting to the student's level of motivation/interest and proficiency:

> A good tutor is able to accurately monitor and model each trainee's knowledge state. That is, the tutor is able to determine what each trainee knows, does not know, and knows incorrectly...The tutor's knowledge of the trainee's level of knowledge extends in time allowing the tutor to make predictions about instructional level and pacing. [Ps91]

A tutor must therefore be able to marshal various resources and skills when interacting with his students. To do this, he must avail himself of an extensive set of knowledge resources: books, graphics, films, pre-prepared courses of instruction; and the skill to use these appropriately. Hypermedia networks can both represent these resources and provide concrete connectivity between abstract concepts that will coordinate student topic selection and learning needs with instructor topic selection and teaching points.

One problem that existing ITSs has faced is that tutorial module responses to student queries tend to be at the wrong level of detail for an individual learner: they offer either too much or too little information [Ps88, Ps91]. The unique interactive and information-presentation powers of hypermedia could extend the tutorial module's ability tailor instructional

sequences to individuals. A hypermedia system could also integrate instructional module resource needs with the resource networks represented in the expert/knowledge representation module.

3. Student-diagnostic module

The primary function of this component is to represent a student's level of knowledge accurately, and to do this dynamically--i.e., during a lesson. This "automated dialogue" reflects the communication process between learner and human tutor, and is the theoretical basis for a focus on communication knowledge, whose goal is to have computer and student act as partners in learning how to manipulate content and relationships in the transfer of knowledge [Ba91, Sw91, Wf91].

A student's knowledge has been represented as a semantic network [Br82, Bu91, Cr89, In89, Ps91, Sl82]. Hypermedia links and nodes could support this representation. An acknowledged shortcoming of existing ITSs that hypermedia may eliminate is found in the knowledge representation and interaction with the student, which tends to be inflexible and rigid:

> The system assumes a particular conceptualization of the domain, thereby coercing a student's performance into its own conceptual framework. None of these systems can discover, and work within, the student's own conceptualization to diagnose his "mind bugs" within that framework. [Sl82]

In hypermedia systems, the locus of control centers on the user. A student could actively determine which "learning links" to navigate through, based on his needs at the time of the tutorial session. In addition, individual student profiles can be analyzed more easily when associated with links

established by the expert module and tracked by the tutorial module, for example: "Given a student model that may be a huge semantic network of the appropriate knowledge, the [tutorial] model may analyze an individual trainee's knowledge network into specific trees or grammars to decide which nodes are ripe to be used to attach new information to make the structure grow...." [Ps91]

4. Interface module

The primary function of this module is to present information to the student both as required to illustrate teaching points and as determined by the student's interest/selection of hotspots. Most importantly, the student should not have to learn a computer interface to learn a new concept or skill. "Students cannot learn from a program they cannot use." [Ps88] Hypermedia provides an intuitive structure for presentation: the mapping of hotspots that appear on the interface screen/text/graphic surface to internal knowledge bases that can be activated to bring up further display selections or the stored materials that have been linked to the hotspot.

A key interface issue is the level of interactivity provided in an ITS. Current research concludes that is should be high, and yet, in existing ITSs "user interaction is still too restrictive, limiting the student's expressiveness and thereby limiting the ability of the tutor's diagnostic mechanisms." [Sl82] The interface must be sensitive to the student's proficiency level and background knowledge. It must provide an entry to the content domain, and be flexible enough to accommodate a student's general skills and/or developmental level. In a hypermedia interface, the user controls the both the level and presentation of information; likewise, a student using a hypermedia-based ITS is empowered to control and even initiate certain levels of instruction.

5. Instructional Design Aids

> It is fairly clear that ITSs cannot replace human instructors in the foreseeable future. Instead, ITSs provide ways of augmenting and extending the power of teachers. ITSs can serve as instructional amplifiers that let teachers personalize instruction more than they now are able to. [Ps88]

Hypermedia concepts can be extended beyond the ITS architecture to support human instructors develop a programming methodology for sequencing lessons and evaluating students. The development of an effective learning environment requires a well-designed knowledge representation structure. "The representation must deal with such critical questions as whether to focus on content or task structures; and provide a way of modelling a trainee's knowledge structures and misconceptions." [Ps91] A hypermedia environment, by virtue of its network structure, allows an ITS author to view a knowledge representation database from any one of multiple perspectives--graphics, sound clips, video clips, text. A hypermedia environment can represent "content" (declarative knowledge), as well as "task structures" (procedural knowledge), which an instructor can use to associate, differentiate, compare, contrast, etc., teaching points in a lesson, or series of lessons. In addition to facilitating a systematic approach to instruction design, an instructor's "toolkit" could reside within the hypermedia-based ITS, which would ensure a consistent interface between instructor development and student execution of lesson plans.

As stated previously, the current ITS design philosophy espouses an interactive, information-rich environment that encourages exploration yet is anchored by specific teaching points as required by specific knowledge domains

and specific learner proficiency levels. The creation of environments that facilitate learning in this way boils down to the following design problem: how to represent extensive quantities of information that exist in a variety of formats and allow guided navigation of that information. A hypermedia system attempts to integrate declarative and procedural knowledge and generate explanations based on user-controlled interaction within an object-oriented framework. The management of communication knowledge through hypermedia offers a stepping stone to further research in cognitive theory as well as artificial intelligence. Ultimately, a hypermedia system could function as the "intelligent" instructional strategist long sought after in ITS design and implementation.

III. COGNITIVE THEORY, EFFECTIVE INSTRUCTION AND HYPERMEDIA

A. CONSTRUCTIVIST COGNITION

Theories of cognition seek to describe and formalize human learning processes in an attempt to improve instruction among humans as well as to construct machines that can "learn" and behave "intelligently" [Br82, Ha88, Ps88, Sl82]. The basic assumption driving research in cognitive science is that learning requires the construction of mental models about the knowledge and/or skills to be acquired. Constructivism is a widely supported branch of cognitive science which espouses the idea that these mental models must be actively constructed by the learner in order to become truly retained and retrievable knowlege [Sc91, Sd91]. Other major assumptions of constructivism maintain that this active learning should be situated in real world contexts (i.e., simulations), and that, although the knowledge acquired is based on personal interpretations, learning should also be collaborative, since "conceptual growth comes from the sharing of multiple perspectives...." [Mr91] Related to these premises is the understanding that every individual enters a learning environment equipped with different mental models, learning styles, and proficiency levels. Thus ITS developers have consistently focused on "providing individual users with instruction tailored to each individual's specific cognitive state," emphasizing that "both the interface and the evaluation must be seriously designed with a cognitive theory in mind." [Ps88]

B. COMMUNICATION KNOWLEDGE

The notion of communication knowledge, or the development of an "empathetic" partnership between instructor and student, is central to the design and implementation of interfaces that emulate individualized instruction. Swigger [Sw91] defines "a successful communication act" as the "[management] and [manipulation] of content and relationships in order to come to some agreement about object X." Constructivist tenets directly parallel this definition: knowledge is organized in structures (content), and knowledge units are interlinked (relationships) by learners in order to construct mental models. In keeping with the cognitive theory focus of ITS design, communication knowledge seeks to implement more flexible learning environments: "...the evolution of system design is moving toward student-centered, reactive learning environments." [Bu91] Three levels of user-computer communication reflect this evolution in design.

1. Traditional person-computer interfaces

In these interfaces, the computer is a passive partner. The user makes requests and gives commands, e.g., retrieve data, sort data, modify, input, delete. Properly formatted and syntactically correct requests are accepted and executed by the system; interaction is relatively one-sided and inflexible.

2. Traditional CAI interfaces

In these systems, communication is computer-controlled and sequential. In a typical transaction, the CBT/CAI system solicits information from the student, then evaluates student responses based on comparisons with its "expert" knowledge base. The Computer-Assisted Language Instruction System (CALIS), developed by Duke University's Humanities Computing Facility, which poses

multiple choice questions to students following reading comprehension drills, is one example [Dk91].

3. ITS

Theoretically, ITSs must be able to model various levels of communication and provide responses that adapt to the student's level of cognitive development. These "student-centered" systems convey the meaning of the conversation through "explicit operators that are supported by specific functions and relationships." [Sw91] These operators are physically available to the student and can be manipulated like real world objects. The student is actively engaged in the construction of his knowledge while being guided by the system-supported relationships and functions, which represent both the content and the context of the conversation [Sw91].

C. INDIVIDUALIZED INSTRUCTION

Indivisibly coupled with theories on learning and cognition are methodologies regarding the nature of teaching and instruction. For every learner who requires "individualized instruction," there exists a different category of knowledge to be imparted, and a different strategy by which to communicate that knowledge. The theories and methods of teaching investigate the following question: how can the best aspects of each approach, since each is suited to different learning situations, be integrated in ITS design to provide "effective" instruction?

In discussing traditional teaching methods and the teaching architectures that embody them, Schank and Jona [Sc91] evaluated and codified the most effective aspects of each approach, concluding with a proposed prescription to combine some methods into a general ITS architecture.

1. Sponge method

This traditional pedagogical strategy assumes that a learner is a passive "absorber" of knowledge. Primarily used in the past due to high student-teacher ratio and lack of books for every student, this method does not account for knowledge that a learner already has, or for a learner's ability to evaluate and question what he is "absorbing." The "trick" to making this approach work lies in motivating the student to listen. It can be necessary when providing a student with the declarative knowledge (i.e., facts/concepts he must internalize) he needs to acquire further knowledge, develop certain skills, or perform certain procedures.

The CBT architecture that embodies the sponge instructional strategy is traditionally known as a "page-turner." [Sc91] A student reads screens of text--an electronic book--and answers questions about topics that have been "paged through." The questions are in multiple choice or fill-in-the-blank form, and only serve to evaluate how well a student can recognize question-answer formats and respond in a properly formatted way. This architecture has failed to improve the value or effectiveness of training delivery systems [Sc91, Ge87].

2. Cognitive apprenticeship method

This teaching strategy patterns instruction after classic apprenticeship techniques: i.e., a student learns by watching a mentor perform a task, then attempts to perform the task with the mentor's guidance. The instructor is a "coach" who prescribes standard procedures to solve problems. The problem inherent in such a strategy is that, without some understanding of the fundamental background knowledge required to perform a skill, procedures quickly become rote responses, testing only recognition of typical problem

patterns. There is potential for the apprenticeship method to degrade to the sponge method. The ability to follow procedures is adequate in situations that require standard procedures, such as emergency procedures, or research methods, or computer I/O protocols. However, they do not lend themselves well to solving problems that cannot be broken down into sub-skills, such as understanding how to apply the basic laws of physics required to solve kinematics problems.

This teaching strategy is often implemented in expert systems and expert system-based ITSs. Termed a "case-based" teaching architecture, this architecture requires an extensive knowledge base as well as a framework of situations that would motivate students to explore the cases and the procedures used to solve the cases. Ideally, this architecture could be used when a student must build, design or create something on an ITS, using the ITS's knowledge base.

> The task...is to teach the student what he or she needs to know, or might consider while doing the task, at precisely the points in the task at which the student becomes interested in knowing this information. This information should be presented in the form of stories. [Sc91]

3. Artist and Exploration Method

This instructional strategy focuses on providing student with environment that encourages student creativity, or "discovery" on his own. Based on constructionist theory, this method requires close monitoring and individualized attention by instructor, to ensure that the student acquires the requisite knowledge to explore certain subjects. Without a supportive guide or

general framework of teaching points, a student can be easily discouraged from exploring on his own.

This teaching strategy can be implemented with a hypermedia knowledge base architecture, which is sometimes called a "directed exploration" approach. An instructional designer who chooses to implement this strategy must ensure that the student's navigation through the knowledge framework is indexed by teaching points so that a "lost in hyperspace" feeling does not occur during a particular learning session [Ni90]. "Lost in hyperspace" refers to the frustration a student feels when asked to navigate freely through knowledge bases without learning objectives to anchor him, or hints about interesting exploration paths to follow [Ni90].

Incidental learning architectures could also support the artist/exploration method. Incidental learning refers to knowledge acquired "in passing" and involves gaining knowledge by completing activities that present or include the knowledge to be acquired in implicit ways. An incidental learning architecture "involves the creation of inherently fun tasks to learn otherwise dull material." [Sc91] For example, in watching a movie about a foreign country, in a foreign language with subtitles, a foreign language student reinforces vocabulary, idiomatic expressions, or conversation conventions. The language learning is incidental to the cultural presentation, and vice versa. Hypermedia-based multimedia design complements the ideas behind incidental learning architectures and could be used to effectively implement the exploration teaching strategy.

4. Research method

With this teaching strategy, a student is asked to research a topic and present a report of their findings. If the student is interested in the topic assigned, the motivation to learn is high, which would engage a student to become an active participant in the learning process as well as provide an attitude to learning that facilitates retention of knowledge acquired. An important aspect of this method is that the activities required for the presentation of research be active in nature--not a mere "book report." [Sc91] As with the exploration method, hypermedia-based directed exploration and incidental learning architectures provide an implementation strategy of this instructional strategy.

5. Argument method

Essentially, this is a variation on the traditional research method, which requires that a student actively test and evaluate summaries of research. With the argument method, a student must adopt an opinion on an issue and defend it. This method is supported by the communication knowledge approach to instructional design [Sc91, Sw91]. Simulation architectures embody this approach in that students can test their theories in a "real-world" context. For example, "...if you want to learn about investments, use the market simulator...if you want to learn how to deal with a client, use the client simulator...if you want to learn about Napolean, talk with the Napolean simulator." [Sc91] The simulation architecture is most effective when the subject to be learned is task-oriented, and situations can be presented that require application of the tasks. By learning about, performing and completing the task (or task structure), a student is actively engaged in testing, evaluating and providing rationale for his ideas, opinions and performance.

D. INTERACTIVITY

> The responsiveness of the computer, the immediate and individualized testing and feedback capabilities it can provide, demands a new understanding of interaction on the part of the designer. [Sg84]

Constructivism in cognitive theory, an emphasis on individualized instruction in teaching theory, and the communication knowledge that combines them, have led to a new watchword for the development of ITS architectures: Interactivity [Bu91, Sw91, Wf91]. Interactivity can be defined as a "mutual and simultaneous activity on the part of both participants, usually working towards some goals, but not necessarily." [By91] Interactive systems can thus be viewed as applications of communication knowledge. Five corollaries support this definition and serve to characterize interactive systems [By91].

1. Interruptability

This corollary reflects that interaction is simultaneous and mutual between computer environment and user, yet interruptible by either; e.g., a student can query the system, while the system can customize responses based on user input.

2. Graceful Degradation

This corollary helps maintain a dialogue between computer and student. When some point in communication is unclear, or requires amplification, the truly interactive system is flexible enough in response generation to say, "could you repeat, or clarify" and "wait a moment while I look for what you have requested."

3. Limited Look-ahead

This corollary emphasizes that a system expecting to be interrupted often should not develop a pre-programmed plan of communication. Ideally, interactive systems should be opportunistic; i.e., communication processes should not have to occur in a strictly pre-determined or hierarchical way--they should be able to occur asynchronously, as initiated.

4. No default

In contrast to traditional systems which provide default paths to support comunication, truly interactive systems enable users to design paths "on-line." This idea is supported by the corollaries "limited look-ahead" and "infinite database."

5. Impression of an infinite database

The knowledge base must be extensive, or at least provide the illusion of many search choices. Otherwise, interaction will be reduced to one (or a few predictable) search procedures--in short, a pre-determined communication plan.

The new emphasis on user-computer interactivity, coupled with the evolution of user-computer interfaces previously described, are both causes and consequences of a new perspective regarding the design not only of ITSs, but all artificial intelligence oriented knowledge-based systems.

> Traditionally, the most widely understood goal of artificial intelligence has been to understand and build autonomous, intelligent, thinking machines...A more important goal is to understand and build interactive knowledge media...or cooperative problem-solving systems. [Fi90]

Cooperative problem-solving systems differ from classical expert systems in that the human is more an active agent and participant. Different levels of interaction are chosen depending on the user's knowledge, the user's goals, and the task/knowledge domain [Fi90]. Ideas regarding cooperative problem-solving systems reflect the context-based content framework necessary to implement communication knowledge theory in ITSs.

E. CONNECTING THE DOTS: HYPERMEDIA LINKS CONTENT, LEARNING CONTEXT AND LEARNER

How can hypermedia-based interactive multimedia systems support the integration of all of these theories? Ultimately, each strategy is concerned with empowering the learner. The overall goal is to create an environment that facilitates learning and is custom-made, or customizable to each individual learner. The link structure of a hypermedia information network is determined by the user as he navigates throughout it. Hypermedia is customized as it is constructed. In addition, hypermedia is a natural paradigm for implementing communication knowledge theory--for linking concepts.

The common thread running through each of these theories leads to a basic framework upon which to design a hypermedia-based ITS: apply the exploration strategy with appropriately-timed coaching. What is meant by appropriately timed coaching? Consider a "Jungle Gym" analogy. This metaphor is based on an education philosophy called "web" learning and teaching, which espouses the idea that "students develop the cognitive structures that reflect the curriculum. The structure of web teaching is a framework of general concepts that is anchored in existing knowledge and that serves to support more detailed knowledge." [Ha88] The limitation encountered in a web teaching approach is

that it provides a static framework for curricula but does not involve dynamic formulation/reformulation of lessons, or knowledge presentation during an interactive (dynamic) tutoring session. In short, the web teaching concept does not solve the dynamic communication management questions: down what paths should a lesson be redirected in response to unexpected questions/answers from the student? How can a student profile be developed and maintained both during a tutorial session and from lesson to lesson? The "Jungle Gym" analogy prescribes a static framework and potential for dynamic interaction as follows: major teaching points of a lesson are explicitly designed with authoring tools and form a network of information nodes; and during a learning session, a student can navigate freely througout this anchoring framework of concepts via hypermedia hotspots and buttons ("show the definition," "take me here," "show a picture," "let me hear it," "bibliography links").

F. PRESCRIPTION FOR IMPLEMENTATION

Educators should capitalize on authoring tools provided by commercially available application developer environments. Multimedia digital publication tools patterned after "what-you-see-is-what-you-get" (WYSIWYG) word-processors provide cut, paste, append, format options for a variety of media beyond text representations. With these resources, educators can implement appropriately timed coaching via explicitly programmed (linked) tutorial suggestions. The lesson templates created by educators can form the prototype hypermedia networks and be used to gather empirical data regarding ease of use, improvement of time management, intuitive interface, etc., for future development of intelligent tutorial modules.

Further, the tutorial templates that are created can be developed into standard knowledge transaction shells--meta-rules for the selection and sequencing of various types of learning objectives. For example, vocabulary definitions, which are a form of declarative knowledge (facts), could have automatic link attributes to a language dictionary database, a graphic descriptor database, an audio pronunciation database, as well as annotations created either by instructors for use in a lesson, or by students for "pasting" to study notes. Hypermedia provides the design methodology; interactive multimedia provides the implementation strategy.

The most exportable, portable, cost-efficient, widely available and compatible delivery platform is the multimedia personal computer (MPC)--the answer to the ultimate video game [Er91, Sc91]. Students interact with knowledge base and lessons (tutorial module) via graphics buttons, document and screen hotspots, and sound clips that are controlled by hypermedia links. This user-oriented design reflects the interactivity principle of "graceful degradation:" some hypermedia systems prompt the user for responses, then tailor the information based on the user's responses; others default to online path-planning, by which the user chooses the topics, level or detail, and order for navigating through information bases. This system could also incorporate facilities to monitor a student's navigation process in order to gain insight on his educational background and misconceptions. Student profiles can be constructed by shell programs that record an I/O "study trail." These data can be stored in student working directories, which can then be monitored by an instructor. Research options in the student diagnostic area include production-rule-based comparisons between expert knowledge representation and student-profiles. This

methodology approaches classic artificial intelligence goals: that of the omniscient tutor.

IV. MULTIMEDIA/AI INTEGRATION AND COMPUTER-ASSISTED LANGUAGE LEARNING

A. INTERACTIVE MULTIMEDIA

Currently, most of the multimedia systems commercially available emphasize presentation features: slide-shows, videos overlayed with graphics and music, and text supported by animated graphics [Er91, Ph91]. Instructional designers seek to extend multimedia's presentation capabilities in forums for communication and collaboration. Learning requires interaction.

1. Brown University's InterMedia: OOP + Hypermedia

> It is important to facilitate the observation and creation of connections and relationships among ideas, concepts, events, and people. Our first general aim, therefore, is to provide software ...that will allow professors to create webs of information and that will allow students to follow trails of linked information, annotate text and illustrations, and communicate with other students and the professor. [Me86]

InterMedia is an object-oriented hypermedia system and applications development framework that is part of Brown University's "Scholar Workstation Project." [Me86] The primary goal of this campus-wide project was (is) to introduce "general-purpose scholarly software" into university courses via networked workstations. Emphasis was placed on observing the impact of hypermedia on both teaching and research endeavors. The design team believed that a "highly functional hypermedia system" could be developed based on present workstation technology, using improved graphics and networking

powers. They hoped to create this system using object-oriented software technology that would facilitate the addition of new features as well as allow educators and researchers access to a collection of software tools with which they could build and link documents and courseware aids.

Originally, the design team chose the Sun workstation and IBM RT PC as the development vehicle and delivery vehicle, respectively. Beyond the underlying operating system and hardware requirements, the major components required by InterMedia were a user interface "toolbox" and a higher-level applications framework. The design architecture specified that an applications framework be built on top of the user interface toolbox. This applications framework would provide higher-level data abstractions and a working "generic" application from which the developer can build his/her software [Me86]. The team decided that the CadMac Toolbox, a version of Apple's Macintosh Toolbox implemented in C on a UNIX base, along with Apple's MacApp programming language, as the software development environment that would best fulfill their requirements. InterMedia (version 3.0) now runs under A/UX (Apple's version of UNIX) on an Apple Macintosh and uses the familiar look and feel of the Macintosh graphical user interface [Ka90, Ph91].

The team defined several educational objectives that their system would realize:

> 1) enabling *connectivity* of information stored in the system's database in addition to permitting communicaton of ideas/questions between students and teachers, teachers and teachers, even students and students;
>
> 2) facilitating *audiovisualization* of concepts by exploiting current graphics and audio system capabilities;

3) encouraging *exploration* by students of a well-integrated knowledge database.

Given this set of educational objectives, the expressibility of OOP languages seemed to offer the InterMedia team with software support features their system required. OOP is founded on objects, object classes and relationships among and between them [Ne90]. This fundamental OOP concept, combined with the natural network structure of hypermedia, provided the framework necessary for the realization of the project's goal for *connectivity* of related course materials. With regard to *audio-visualization* capabilities, the various media objects that could be supported by hypermedia enhanced a "student's ability to visualize and perceive complex and/or dynamic phenomena" (e.g., cell division, music notation and sound, mathematical surfaces, programming structures) [Me86]. Finally, hypermedia, by definition, facilitated *exploration* of an information-rich environment so that students might not only discover ideas, themes and facts *on their own* but also become inspired to actually synthesize knowledge--to add *their own connections* and *create original materials* within the system [Me86, Ka90].

In addition to the above educational objectives, the team defined the following technological objectives that InterMedia should achieve:

1) ensuring *extensibility* of the system from its initial capabilities to applications which could be added with relative ease by future researchers/designers;

2) developing *reuseable* code;

3) providing *consistency* across applications (i.e., a standard user interface for both graphics and word-processing applications);

4) aiding *understandability*, which the design team defined as "a uniform way in which individual developers could understand the data structures and procedures in the system" [Me86]; and finally,

5) permitting *modularity for parallel development* of separate applications such that developers could both test code without interfering other developers and integrate their modules into the overall system with minimal effort.

In theory, OOP is a paradigm ideally suited for modular system design: the details of program units developed and modified by different people can be strictly encapsulated (or hidden), which can alleviate the interaction and dependency problems of large design teams [Mi88]. The OOP notion of data structures as objects, with certain defined behaviors and inheritance hierarchies whose implementation details are encapsulated from individual developers also fulfilled the InterMedia team's *understandability* and *consistency* objectives--ultimately, a developer need only concern himself with how his unit interacted with the rest of the system--each module could have a "uniform external interface." [Mi88] The basic OOP concept of inheritance, or code-sharing, supported InterMedia's requirements for extensibility and reuseability [Ne90].

In developing the entities by which users would interact with InterMedia, the team introduced the concept of information *webs*, which could be linked based on user selections. Conceptually, webs correspond closely with hypermedia networks: "A web should be thought of as a database that contains both references to a set of documents and the links associated with those documents. Two webs might reference the same documents but have entirely different sets of links." [Me86] Additionally, because users would be given the power to update, or otherwise modify the same body of documents or blocks of

information simultaneously, a network file framework (that would be transparent to users while allowing them access) would need to be developed. This network framework would maintain access control for the update of documents and links and prevent concurrency problems. The OOP paradigm--objects with uniquely defined behaviors and inheritance links--together with the hypermedia-based mapping between internally represented entities (sound clips, images, documents, etc.) and objects presented (visually and aurally) by the interface, provided an ideal framework for such system specifications.

Another InterMedia system specification that would benefit from application of OOP principles such as inheritance and encapsulation as well as hypermedia linking concepts was the application building feature:

> InterMedia should provide basic levels of functionality for applications to operate...so that application developers can concentrate on developing those features unique to their application. Such common features should include standard libraries for handling windows, menus, the linking and blocking of functions, interactions with the desktop, and like functions...InterMedia should provide developers with building blocks that implement ...common functionality and are useable by all applications developers with little or no modificaton. [Me86]

To support their consistency and connectivity objectives, the first requirement the team imposed on the system was that it provide a "direct manipulation interface" for all operations [Me86]. Hypermedia is exactly that: the fundamental principle behind the "what-you-see-is-what-you-get" (WYSIWYG) word-processing metaphor [Ph91]. Encapsulation would support the requirement of developers to create applications that were independent of yet easily integratable

into the overall system. The concept of object classes could be extended to support the development of InterMedia's standard libraries of user-interface functions. Finally, inheritance would support reuseability of application-development code among individual designers.

Because one of InterMedia's specifications called for simultaneous access/revision to the information webs, management of updates and maintenance of data integrity was a concern. Consequently, the team decided to store InterMedia's information in a single, system-wide database: the InterMedia Database. The interface goal declared by the design team was that database procedures be hidden from applications developers, who could only access information through InterMedia Block, Link, and Web classes--the class methods would perform all the linking, retrieval and updating control operations (thus enforcing encapsulation). These class groups provided an external interface for an application developer. There were two primary "building blocks:" the text building block and the graphics building block. The classes defined in the text building block contained methods which provided conventional wordprocessing features such as reformatting, style-defining, scrolling selecting. The graphics building block provided support for creating, drawing, painting, cutting, positioning, resizing and viewing graphics objects. The building blocks' level of InterMedia provided the foundation for the final layer--a set of applications' components: the folder directory system, the graphics editor, the text processor, the timeline editor, the scanned image viewer, and the web maps. These components are the hypermedia representations of the multimedia objects that can be used to develop InterMedia educational materials.

The InterMedia model addressed two instructional design problems: presenting and representing teaching points using connectivity and visualization of concepts [Ka90]. Because it is an environment in which hypermedia functionality is shared by all applications (both system-based and user-created), it is an effective tool for integrating learning across many different subject areas. "The ability to link information together provides and intuitive means for students to relate events and ideas in different contexts." [Ka90]

2. Perseus

> Allowing fluid movement among disparate date, hypermedia databases ideally suit the study of ancient cultures, where evidence ranges from literary texts to archeological sites to individual objects. [Cn91]

The Perseus project at Harvard University provides hypermedia support for the study of ancient Greek literature, archeology, and history, using CD-ROM storage for its immense hypermedia database. The project collected as many kinds of data as possible that dealt with ancient Greek civilization from the eighth century B.C. to the late fourth century B.C., in an attempt to afford students of antiquity access to as many kinds of information (i.e., "multimedia") as possible. The hypermedia database includes source texts in Greek ("primary" material), English translations, essays, maps, motion video and still images representing Greek art and other archeological materials [Cn91, Ma91, Ni90].

The educational philosophy on which the Perseus Project design is founded proposes the following pedagogical principles:

> 1) Learners construct their own knowledge, and instructors organize the context for such constructions;

2) Access to primary materials facilitates independent synthetic thinking;

3) the study of culture requires multiple modes (i.e., multiple points of view and sensory inputs--sound, visual) of human experience.

The design team also cited several mechanical advantages of hypermedia over traditional student study and research methods:

1) the broad range of source materials (sound, text, graphics) available for exploration would provide a richer learning experience;

2) the on-line availability of primary sources as well as interpretive essays facilitated (especially in speed and access convenience) the literature/cultural searches that students were required to do when conducting research;

3) the immense size of the on-line database provided students with the potential to find more potentially relevant passages or graphics support for their own research hypotheses. The hope was that, by combining information resources with a pedagogical structure via a hypermedia database, an "intellectual infrastructure" that would enhance, enrich and improve education for students of ancient Greece. [Cn91, Ma91]

To evaluate the effectiveness of presenting educational materials through this hypermedia database, Greek scholars and educators evaluated the essays of two groups of students: a control group that did not have access to the Perseus database, and an experimental group that used Perseus. Results of these evaluations showed that there was no significant difference in the quality of the student essays on any criterion measures between Perseus and non-Perseus

students [Ma91]. However, there seemed to be incidental learning advantages: although having more citations to choose from did not seem to improve the quality of essays, knowledge acquisition skills (how to reference material, how to evaluate it) improved, and there was more collaboration between students. There were different levels of criticism for some of the reference materials: literature students felt that the graphics references provided were "great," while archeology students, whose studies depended on accurate representations of artifacts, felt that they could be improved upon [Ma91].

B. COMPUTER-ASSISTED LANGUAGE LEARNING: PASSIVE PAGE-TURNERS VS. INTERACTIVE CONTEXTS

Many of the programs now available for foreign language instruction appear to be computerized versions of the traditional grammar-translation based textbooks [Cr89, St91]. Although CBT has been employed in foreign language instruction extensively over the past several years, the courseware produced thus far has been characterized by the following limitations: text-based software on low-powered systems; instructional modules which implement quiz-based teaching methodologies instead of interactive, contextualized learning environments; and limited use of multimedia [Af90, Dk91, St91].

The majority of foreign language courseware has been developed on low-powered MS-DOS microcomputers which required large amounts of programming for small amounts of lesson material. Just as every foreign language has specific syntax, grammar, and cultural elements and rules, every computer-based lesson required a different tutorial program to support sequencing of instruction and representation of the knowledge base for that target language. Beyond the most basic language drills, the implementation of a

standard knowledge representation scheme across many different foreign languages seemed impossible to achieve. Consequently, the lessons produced were very simple and repetitive. Students were presented with text and responded with typed inputs. Interactivity was non-existent, so that any type of the communicative, contextualized learning enjoyed in the classroom with a human instructor and classmates was not reinforced. Computer-based lessons were not meaningful to students; they were rote memorization drills. Students were unmotivated: textbook exercises provided the same rote drill practice and required no computer skills to comprehend and complete.

The one feature that has distinguished early CBT lessons from traditional textbooks was the ability to provide immediate answer judging. A student could receive immediate feedback on homework, both in answers to specific questions as well as in overall performance grades. The end result of this methodology was the emergence of a quiz-based, "teaching by testing" pedagogical approach [Bm91]. This approach, "Computer Assisted Language Learning" (CALL), is characterized by the presentation of a variety of reading comprehension examples followed by a series of drills which include multiple choice, CLOZE (essentially, fill-in-the-blank questions for foreign language learning), short answer, and grammar transformation exercises. Students must work through the exercises sequentially, and the range of responses accepted by the system as "correct" are limited by the range of values explicitly programmed as "correct" answers. Ultimately, students become proficient, formatted response generators, and their evaluated performance almost completely sidesteps the development of listening, reading, speaking, writing and translation skills which are the intended outcomes of foreign language learning. Experienced linguists are not motivated

to practice or refresh their skills; the lessons are boring and frustrating in their inflexible sequencing and limited range of acceptable response inputs.

With regard to multimedia use and implementation possibilities, computer technology has only recently allowed multimedia to be easily incorporated into CALL lessons. Sufficient mass memory storage to include extended audio and video recordings has been a problem. Technological advances in interactive videodisc hardware have not been used to their full advantage due to high costs of video production. As a result, multimedia-based CALL materials have not been developed beyond what has been available through traditional audio-visual technologies.

Tutorial applications that were cumbersome and difficult to program, CALL techniques that proved tedious and unmotivating, and the lack of commercially available multimedia technology severely limited the effectiveness of teaching foreign languages with the help of computers. However, the integration of object-oriented and hypermedia data structures and design offer promising new perspective in CALL development approaches. In addition, the commercial availability of multimedia extensions to existing systems has potential to repurpose existing technical resources--e.g., audio tapes, videotapes from satellite broadcasts--into interactive formats more conducive to teaching and improving the essential foreign language skills of speaking, listening, and translating.

The traditional language laboratory system consists of audiotape listening and pronunciation exercises. A multimedia language laboratory program will turn audiotape listening into an interactive recording session. More than any other type of learning activity, the study of foreign languages depends on the

presentation of different modes of communication. Cognitive theory asserts that the number of modalities of information (still photos, moving pictures, sound clips, textual data) by which a concept or body of knowledge can be presented influences the concepts learners form and their ability to execute these concepts [Bg85]. Because concepts are formed using various sensory and mental inputs, conceptual links that are created between and among various visual, auditory, and motoric data can all become part of the same knowledge base. Speaking, listening, translating, interacting and collaborating with people who not only have different languages which to communicate, but also have different cultural backgrounds, require skillful integration of various modalities of information. Key aspects of foreign language learning that can be enhanced with hypermedia-supported CBT follow.

1. Memory

Interactive translation exercises will enable students to quickly master new vocabulary and grammar contextually. Translation exercises will be constructed by instructors. System feedback and prompts will be initiated by student selection and keyboard input. Consequently, *passive listening drills become active learning environments.* By implementing hypermedia strategies, available target language audio and written material will be integrated with grammar notes, translations, cultural information, cross-reference vocabularies to provide a truly interactive "discovery" environment for students.

2. Comprehension

Authentic target language source material (audio and text) will be followed by interactive comprehension questions with instant replay of relevant portions. Pre-exercise overviews will highlight selected vocabulary, grammar,

and cultural features. On-line reference works (linked via hypermedia "language webs") will assist students in reading or listening to new material from extra-curricular sources.

3. Production

On-line bilingual dictionaries, thesauruses, and other reference material will help eliminate the barriers that normally inhibit second language writing. A student who must translate and create original work in a foreign language using the extensive primary source tools available in the digital storage of a computer is immersed in a multi-modal, contextualized learning environment. These language laboratory simulations will build a communication knowledge base for a linguist that could prove invaluable when required to produce translations and compositions on-the-job.

4. Testing

Major areas of reading, listening, writing, and speaking and their component skills of translation, pronunciation, spelling, comprehension, composition and dialogue can each be measured more accurately with computer-based lesson plans. Each skill can be evaluated for accuracy in vocabulary, form and syntax. More importantly, performance speed relative to native speakers can be evaluated in student sound files. Multimedia CBT offers a student dialogue practice and provides an evaluation context which mirrors the skill being tested. Instead of reading a story and choosing responses from a number of alternatives, or "filling-in" vocabulary sentences, a student must produce language--verbally and in writing.

The growth of OOP, with its notions of extensibility and reusability, would help eliminate the limitations encountered in standardizing the knowledge

representation (e.g., rules regarding syntax and grammar) of diverse languages. Extensibility would be implemented by representing parts of grammar as prototype objects, or classes. The class or grammar object attributes could be reusable across many different languages, with the syntactical and semantic differences defined when necessary while the similarities in form and structure could be inherited.

V. THE LINGUIST WORKSTATION

A. FRAMEWORK FOR DEVELOPMENT

"The Linguist Workstation" is the suite of software and hardware that will provide for the delivery of computer-based foreign language learning at the Defense Language Institute (DLI). The primary intent of the workstation design is to exploit an array of multimedia resources now being incorporated with computer systems. Its hardware capabilities are based on standards established by the National Cryptologic School (NCS) for the delivery of Computer Based Training (CBT) [Nc91, Ns91]. The interface to the workstation employs state-of-the-art software with presentation capabilities such as audio and video multimedia supported by hypertext and multiwindowing. Future plans include the application of AI techniques such as the incorporation of natural language processing and voice recognition capabilities, and adaptive instructional delivery based on student modeling and error diagnosis [Bm91].

The development and implementation of this workstation is the primary objective of the BMLC Project. Sponsored by the US Army Special Forces, the mission of the BMLC project is to establish a standard foreign language training system for military linguists in all branches of service [Bm91]. This project is the centerpiece for a long-range modernization plan that was developed by DLI, whose ultimate goal is to create a standard foreign language training system that will encompass thirteen languages. The BMLC project has been approved by the Commandant of DLI and has a projected completion date of 1994. It includes a

technological component as well as a pedagogical component. The functions of each are delineated as follows:

1) *technological component:* installation of 1500 classroom computers with Graphic User Interface (GUI) architecture and multimedia capabilities;

2) *pedagogical component:* development of courseware and the training of personnel (students and faculty) to design, develop, and use the courseware.

The Linguist Workstation is the keystone upon which these two components rest. The workstation is the delivery vehicle with which the BMLC project hopes to centralize resources, activities, and performance evaluations for a multilevel, multilingual language laboratory. As such, it has two major purposes:

1) to provide an *integrated control structure* for course management of a multilevel, multilingual learning lab;

2) to develop a *standardized system* for producing individual lessons. In view of the portability inherent in all forms of computer data, this system has potential for becoming the prototype courseware development standard for computer-based training DoD-wide.

B. DESIGN RATIONALE AND DESCRIPTION

To fulfill the requirement for the integrated control structure described above, the workstation will be presented to users as an interactive learning environment using commercially available application development software (e.g., Apple's *Hypercard*, Microsoft *Windows*). BMLC developers have created a "Linguist Workstation Interface" to free students from the need to learn

another system/skill in order to learn a foreign language. This interface functions as a teacher's aide as well as a student's interactive tutor. Fulfillment of the requirement for a standardized system for courseware design, with multiple languages at multiple proficiency levels, will greatly facilitate the production of lesson plans and performance evaluations. BMLC developers are currently designing and developing a Language Instructor (LI) courseware "toolbox" that will provide word-processing and hypermedia-linking facilities to be used both in the production of LI lesson plans and in the evaluation of student progress.

The notion of a box of course-building tools is based on the use of macro utilities, or shell-script programs. The construction of laboratory management templates is the design methodology being used to develop the Linguist Workstation's courseware toolbox. Conceptually, these laboratory management templates function much like database management systems in order to facilitate course development by LIs and provide easily accessible reference/tutoring material for students. They are program modules designed to control the presentation of material to the student in a pedagogically sound manner [Af90].

The pedagogical basis upon which the template framework is founded is a combination of "exploratory" learning and cognitive apprenticeship theory (or teaching by "coaching"). The premise of these teaching strategies is that the student must be provided with an environment that encourages his creativity, or discovery on his own--that he is an active agent in his own knowledge acquisition. The "coaching" is provided by the template framework, whose structure organizes the context in which the students explore. The learning context is designed as a web, or network of information and options--essentially,

a hypermedia environment. To facilitate navigation through the teaching points constructed, the choice of tasks and goals will be under the control of the *user* (in this case, student), not the system (in this case, computer "tutor").

In accordance with this implementation methodology, LIs will use the Linguist Workstation's toolbox of template instructions to construct a "road map" to control lesson sequencing. These instructions will be activated by student selection: e.g., keyboard input, menu selection, "pointing," "help" menu options, etc. Target language material--both primary sources (e.g., news broadcasts, newspapers,) and summary/interpretative sources (e.g., critical essays, cultural commentaries)--is stored separately (in audio/video files, text files, on-line reference dictionaries) and will be accessed via hypermedia links that have been pre-programmed by LIs using the toolbox of templates. These links will form an on-line vocabulary database that will provide a simple bilingual dictionary entry, information about forms and usage, example sentences and an audio recording of pronunciation. Lesson plan development is being managed by LIs in collaboration with interface module developers.

Hypermedia links can also be used to monitor student progress and evaluate proficiency. The paths which a student chooses to follow on the lesson "road map" will be recorded and stored in a student profile for review by LIs. Paths which a student creates can help to determine misconceptions or understanding. Individualized student histories and generalizable learner "trends" that grow from these paths may provide the empirical data and statistical basis for the systematic development of automated tutoring strategies.

C. HARDWARE AND SOFTWARE

The Linguist Workstation is defined as a Level III Interactive Multimedia system, which means that its combined hardware and software capabilities enable the creation of applications that employ a videodisc player under external control of a microcomputer [Bo91]. Its architecture has been approved for the next generation of CALL courseware at DLI and is in accordance with the emerging Multimedia personal computer (MPC) standard being adopted by industry [Er91, Pe91, Ph91, Wo91]. The adoption of this architecture ensures compatibility and portability between courseware produced at DLI and all MPC systems in use throughout the DOD as will as among academic institutions (e.g., United States Air Force Academy, Duke University). All current and future major military service computer requirements contracts specify MPC compatible computers; however, the multimedia components must be acquired separately [Bm91].

The hardware standard for CBT was established by the NCS to ensure CBT courseware interchangeability and portability throughout the cryptologic training system. The standard applies to all future procurement, development and implementation of NCS CBT [Nc91, Ns91]. The NCS rationale for its standard is that a single operating system and environment will achieve the desired courseware interchangeability. In accordance with this goal, a baseline system was established to ensure that courseware currently under development be compatible with existing systems. This system is defined as follows:

Function:	**Minimum Capabilities:**
CPU	Intel 80286 (8 MHz clock)
RAM	640 Kilobytes (KB)
Display	EGA
Audio	NONE

Storage (internal)	20 Megabytes (MB)
Media Input Format	360 KB-Double-sided, double density (DSDD)
User input mode	keyboard
Operating system	MS-DOS 3.2
Peripheral Devices	printer

The Linguist Workstation represents an upgrade of this baseline that will support the language learning courseware being developed at DLI in response to the industry multimedia standard. This higher performance system is known as *Replicate*, the NCS's Training Delivery System (TDS). The *Replicate* system is designed to both "support the CBT courseware interchangeability standard" and help military linguists "meet the more complex learning objectives inherent in language skill enhancement and proficiency maintenance." [Ns91] *Replicate's* architecture configuration and capabilities follow:

Function:	**Description:**
CPU	Intel 80386 chip (25 MHz)
RAM	2 MB
Display	19" monitor, super VGA
Audio	On-line digital audio card audio delivery system: mixer, headphones, cables
Video	New Mediagraphics VideoWindows Graphic Overlay
Media Input Format	1.44 MB High Density floppy Syquist 44 MB removeable hard disk Laser videodisc
User Input Mode	AT-101 keyboard mouse touchscreen
Operating System	MS-DOS 5.0
Peripheral Devices	Epson FIX-1050 Printer Sony LDP2000 or Pioneer LD-2200 videodisc player

The prototype Linguist Workstations that are currently being developed at DLI use the *Replicate* standard as a guideline and consist of an Intel 80386 CPU with 4 MB of RAM, 80/120 MB hard disk, Soundblaster (or MediaVision ProAudio Spectrum) audio board, VideoLogic DVA-4000 video adapter board and a Pioneer LD-2200 Laserdisc player. To round out the system features, some "master" workstations in the language labs will add a CD-ROM or writable CD as an optical disk storage device.

The primary software development environment is MS-DOS 5.0 and Asymmetrix *Openscript*, the scripting language for the application builder, *ToolBook*. *ToolBook* is a programming environment similar to Apple's *Hypercard*. The workstation interface shell was developed with Wilson WindowWare's *CommandPost*, which runs in Microsoft *Windows* as an MS-DOS resident program [Wi90]. *CommandPost's* macro-command and menu-building capabilities customized the Microsoft *Windows* Graphical User Interface (GUI) into the Linguist Workstation interface and proved more efficient and intuitive than Microsoft Windows for training purposes. Ancillary graphics programs installed in the workstation software environment include Microsoft's *Paintbrush*, and the standard word processing program is Microsoft *Word 4.0*. Additional applications used for developmental purposes include *HyperCard*. Existing courseware that was developed using *Hypercard* will be converted to the MS-DOS environment with conversion programs (such as *ConvertIt!*, a Hypermedia File Format Converter utility that translates *Hypercard* stacks to *ToolBook* books) for final implementation on the Workstation.

The InterMedia system architecture was built upon a MacApp foundation, which defined several fundamental object classes that provided developers with

default values for menu, window, and other application-building features [Me86]. Likewise, *ToolBook* provides an OOP environment that facilitates multimedia courseware development. The *ToolBook* itself is a hypermedia-based interface whose features enable LIs to customize their lesson plans based on inheriting attributes and behaviors with hierarchical links to application templates defined in existing system code. The "system book" templates, along with their inheritance hierarchies, supports extensibility of existing applications. For example, one class of books, "MMWidget.TBK," contains application templates that handle the display of video windows and the video clips associated with them as well as the display of sound frequency files (i.e., spectograms) and the audio associated with them. Because the system interface is founded on well-defined classes of books and book levels, developer understandability of their programming environment is enhanced.

The environment is thus designed to allow component lessons to be developed using any authoring program (or computer program), yet remain fully integrated for student use. This architecture facilitates prototypical courseware development: students actually use, test and provide feedback for lessons as they are being designed and implemented. Its key features include a standard GUI, multitasking, inter-application communications, modular Dynamic Link Libraries (similar to Apple's XCMDs and XFCNs), Dynamic Data Exchange (DDE) protocols, and device-independent multimedia drivers. These software development tools are all based on the Microsoft *Windows* Application Program Interface (API), which ensures compatibility with all MPC systems.

D. DEVELOPING COURSEWARE IN PHASES

The BMLC project managers have devised a three year phased development process to implement the final Linguist Workstation. The primary goal of phase one is to establish the workstation design through evaluation and procurement of the state-of-the-art multimedia hardware and software. Consequently, the acquisition and installation of the various system components has constituted the bulk of the work thus far. The courseware development goal for this phase is to produce a complete set of multimedia resources and exercises for each of the language courses. To expedite the development of these resources, many of the existing educational resources such as videotapes and audio cassettes will be repurposed for use on the workstation: video sequences on videotape and film are being converted to videodisc format, audio tapes are being stored as digital sound files. The LIs for each language group will determine how those resources will be accessed via template instructions. Use of workstation resources at this stage will rely heavily on current textbook lesson sequencing. For example, dialogues and mini-scenarios will be recorded from textbook lessons and exercises for "follow-along" use by students.

The next phase, scheduled to begin early 1992, will undertake a major shift in courseware design. The "hard-coded" tutorial templates implemented during the first phase will evolve into a more comprehensive multimedia database. The text-based database will be enhanced with primary source materials and references throughout which students and LIs can navigate. The data structures that are developed to represent the multimedia resources will be based on object-oriented programming design principles. The hypermedia links will be

constructed, modified, and maintained by an object-oriented database management system.

The final phase, projected for 1993-94, will incorporate voice recognition and natural language technology into the workstation user interface. In addition, tutoring strategies will be encoded using artificial intelligence techniques to allow for a sequence of instruction that is dynamically adaptive to individual learning styles and needs. A student-centered interface will remain the primary teaching feature; however, automated guidance and computer tutor interaction will support navigation through the reference material both to ensure teaching points are experienced and to validate and evaluate language proficiency.

The multilingual, multimedia database will also be expanded in this final phase to include hypermedia links to commercial foreign language applications and on-line resources. These enhancements will establish the workstation as a fully functional linguist's performance support system [Ge87]--i.e., both instructional tutorials and language production tools will reside on one system, for use by linguists of every proficiency level. Some of the linguist production tools to be integrated include: aids to monitor and transcribe audio sources, facilities to provide bidirectional translation of unfamiliar audio or text sources, and foreign language primary sources to support research paper and area study compositions, class lectures, even foreign language multimedia briefings. These tools will motivate language training beyond the classroom: a linguist will be able to maintain language proficiency while on the job.

E. LINGUIST WORKSTATION INTERFACE

The Linguist Workstation interface is a menu-driven environment that gives both student and teacher access to the foreign language learning facilities

installed in the application work level of the system software. Courseware prototypes are currently being developed within this shell, using Asymmetrix *ToolBook* and a *Multimedia Extensions to* Microsoft *Windows* version 1.0 (Beta Release 3) [Ms91]. The interface was developed with Wilson WindowWare's *CommandPost* menu-building language, which runs in Microsoft *Windows* as an MS-DOS resident program [Wi90]. A change was made in the "WIN.INI" file of the Microsoft *Windows* environment to transfer control of the *Windows* GUI from the Microsoft Program Manager to the *CommandPost* environment. Default *CommandPost* menus were customized to create the Linguist Workstation interface.

Essentially, the Linguist Workstation interface is a multiwindow, menu-driven file and applications manager whose appearance is similar to both the Microsoft *Windows* File/Program Manager and the Apple Macintosh GUIs (Figure 3.1). Users can browse freely through the workstation facilities and launch desired applications based on menu selection. Appendix A contains the programming language code for both an edited version of the CommandPost default program ("CMDPOST.CPM") and the program that it calls, "LINGUIST.CPM." "LINGUIST.CPM" generates the workstation interface.

Because system specifications require simultaneous access/revision to language databases, management of updates and maintenance of data integrity was a concern. The interface provides login and password identification procedures to ensure that user workspaces are maintained as separate units apart from the system database. Personal work directories are automatically created for new users based on last name login identifications. System operators have access to *Windows* system facilities based on password verification procedures.

Figure 3.1 Linguist Workstation Interface

A primary system goal is that courseware developers remain free from database access procedures and only concern themselves with the applications development level. Consequently, LI access is limited to word-processing, graphics, spreadsheet and courseware applications (e.g., Duke Unversity's CALIS) at the Linguist Workstation menu level. Only system managers are authorized "super-user" privileged access to the *Windows* Program, File, and System Manager utilities.

The system starts up with a login prompt window by default. Only a system manager can bypass this login procedure. The convention followed for identification is that a user login with his last name. A directory search is conducted by the interface shell program ("LINGUIST.CPM"). The interface is not case-sensitive. If the user has logged on to the system before, a directory will have been created, and the system will transfer all subsequent work to the user's directory. The effect is comparable to a "cd <username>" command in UNIX or MS-DOS. If a "new" user is logging on, or the shell does not recognize the user's name after the directory search, the user will be queried to login a second time, for spelling verification of the user name. If the second directory search does not yield a match, a new directory is created. This directory is the new user's working directory; its name matches the user's login name.

Next, the user will be asked (via dialogue box) to select a language directory within which to work. New users will have only one option: "New." Established users will have the "New" directory option as well as a variety of personal sub-directories from which to choose (Figure 3.1). Selecting "New" generates a dialogue box that requests that the user enter a language. This procedure creates

a new sub-directory with the user-specified language. Any subsequent work is saved in this sub-directory by default. (If desired, work can be saved on floppy disks for back-up via menu selection.) The Linguist Workstation menubar reflects the current working language directory. In Figure 3.1 for example, a user has not yet logged on. This is reflected by the menubar warning: "YOU NEED TO LOGON!"

Once logged on, if the user were to choose Russian as a working directory, the menubar would display the following: "The Linguist's Workstation - RUSSIAN." The keyboard mapping would change to a cyrillic alphabet (the name of the Russian character alphabet), using standard cyrillic typewriter key positions for the characters. A keyboard graphic that reflects the mapping would be available for instant access while a student completes a scenario or problem set, or creates personal study sheets. Because the graphic mapping is consistent with an actual Russian typewriter, students will be learning a lingual applications skill ("touch-typing") while learning the language. If the user chooses not to work in any specific language, the interface would remain in English, with a conventional English keyboard. The menubar would reflect this option with the display: "The Linguist's Workstation - GENERAL" and the working directory would remain at the <username> level.

With the exception of a few language learning applications (Duke's CALIS and the *ToolBook*-based video lesson prototypes described in the following section), these interface utilities are currently limited to English-only file and directory management procedures. Word-processor "linker" functions (e.g., fonts and keyboard mappings) of different languages to the various language directories are still in developmental stages. Once these are completed, however,

LIs and language students alike will have the capability to create documents in their target language.

One of the foreign language shells nearing completion is an electronic version of the basic German texts now in use at DLI. A German keyboard has been created and is being used in "touch-type" practices for students, LIs and linguists conducting skills reviews. Data for bilingual reference dictionaries and on-line help files are being collected, composed, and consolidated for input into a German Grammar Reference. Language courses following the same development process include Korean, Hebrew, and Farci, among others. This work, when completed, will be distributed to language laboratories throughout DLI on CD-ROM discs. Subsequent down-loading of courseware onto individual workstations will be managed at the language laboratory level. Once distributed locally, the courseware will be sent to intelligence specialists at operational commands for use in on-the-job training.

The Linguist Workstation menubar (Figure 3.1) reflects the submenu categories: Applications, Lessons, References, Tutorials, and a Personal Information Manager.

> **Applications** includes access to several commercial applications and the *ToolBook* courseware building utilities. Spreadsheet and database applications will support the management of student lesson profiles for LIs. Inter-applications links between *ToolBook*-based language lessons and the database utilities are being designed using *Windows* DDE's to provide automatic lesson-tracking for both students and LIs. Once implemented, an LI will receive automatic updates on the practice lessons completed by individual students as well as spreadsheet-generated evaluations for

close review. These tools will help identify potential language misconceptions in the early stages of language-learning.

Lessons includes access to both developmental lessons (in *ToolBook* and WINCALIS, a *Windows* version of Duke's CALIS) and implemented lessons (in CALIS). Implemented lessons are currently used for remedial practice in language study labs. The implementation of developmental lessons has been limited to the workstations in the BMLC research lab at DLI.

References will include access to bilingual dictionaries and grammar notes. Plans for additional reference resources include "cultural" notes and area study summary sheets.

Tutorials includes access to applications such as Microsoft *Word* for *Windows* and Asymmetrix *ToolBook*. In addition, based on current feedback from system test-users, on-line help modules are being designed for both courseware developers and student users of ToolBook lessons.

Personal Information Manager includes menu items comparable to paper-based personal planning calendars: appointments, phone numbers, addresses. The information a user chooses to store via these utilities do not relate to language studies and are provided as a convenience.

F. THE VIDEO LESSONS: PROTOTYPES IN GERMAN & KOREAN

Two prototype lessons have been developed using the full interactive multimedia capabilities of the Linguist Workstation. The target languages are Korean and German. Each lesson developed in this environment has the following objective: to provide primary source language material (a foreign film or simulated scenario on videodisc) in various formats and allow the student to

control the speed and sequencing of the viewing session in order to develop the vocabulary knowledge and skills needed to function as an area specialist and linguist in the field. The lessons are designed to accommodate different user levels: from beginning levels to intermediate to practicing linguists. During the lesson, the student has the power to stop, start and replay video sequences; view transcripts of video dialogue in both the target language and English; listen to instructor-created dialogue; and record and playback his own audio recording of transcript dialogues, comparing his pronunciation to the instructor mode. In effect, the lesson interface is designed to provide a feeling of full "immersion" in the target language, and to engage the student with a movie whose viewing modes and sequence are under his control (via buttons to control the player and computer interface).

The lesson template was created using Asymmetrix *ToolBook* and program templates provided in "MMWidget.TBK." "MMWidget.TBK" is a class of *ToolBook* system books containing applications templates that handle the display of video windows and their associated video clips as well as the display of sound frequency files (e.g., spectograms), CD-ROM files, and assorted graphics tools. Figures 3.2 and 3.3 show two pages within "MMWidget.TBK:" Figure 3.2 describes a digital video sequence player while Figure 3.3 highlights various video control panels.

"MMWidget.TBK" exemplifies the ease and speed by which prototype courseware can be developed: its pages contain "widgets" that perform operations in response to user interaction. Each of the graphics objects (e.g., buttons, text fields, windows, switches) is associated with a script that determines its behavior once activated by a user via mouse button clicks or hotkey selection

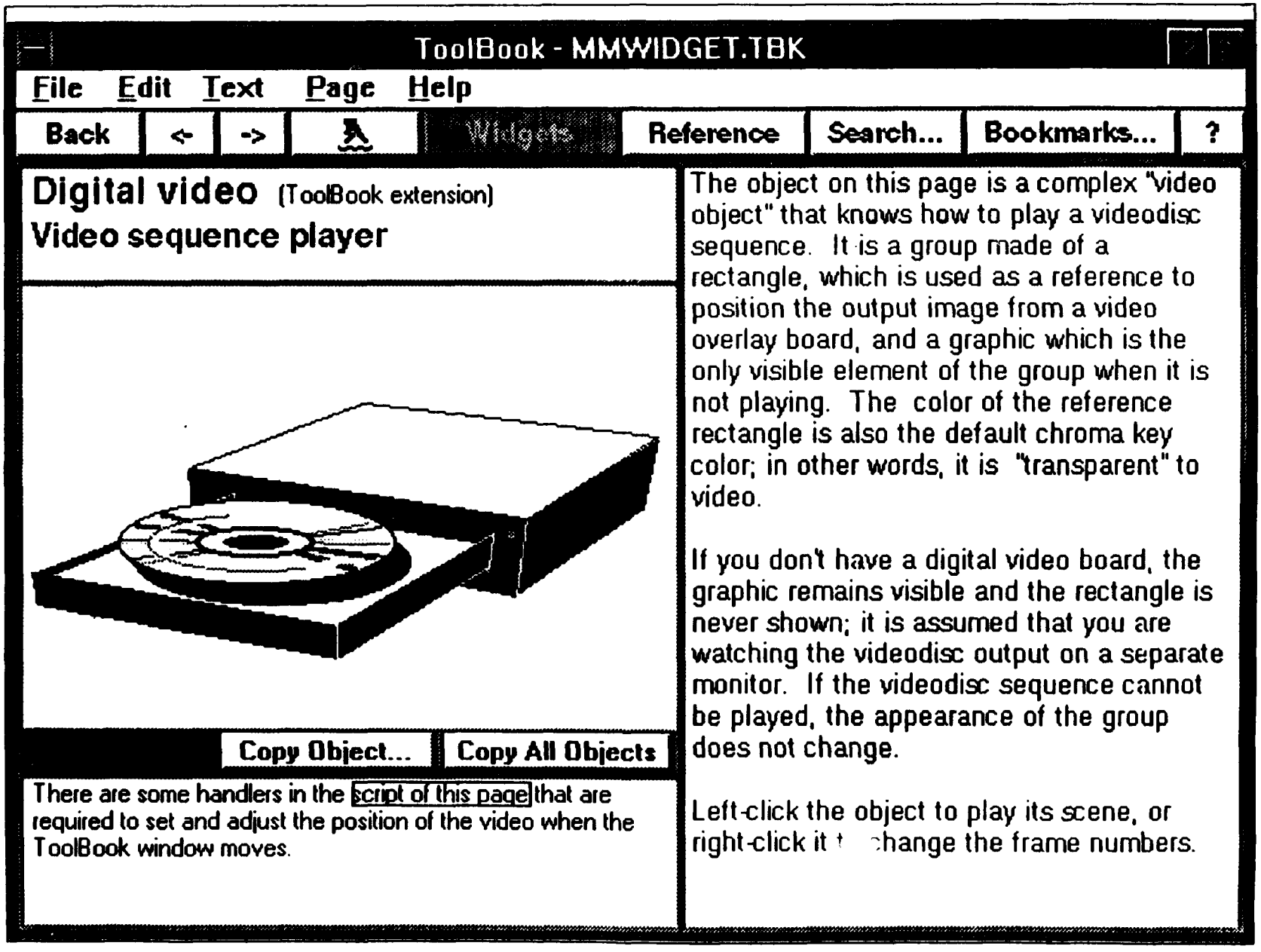

Figure 3.2 Multimedia Video Disk Player

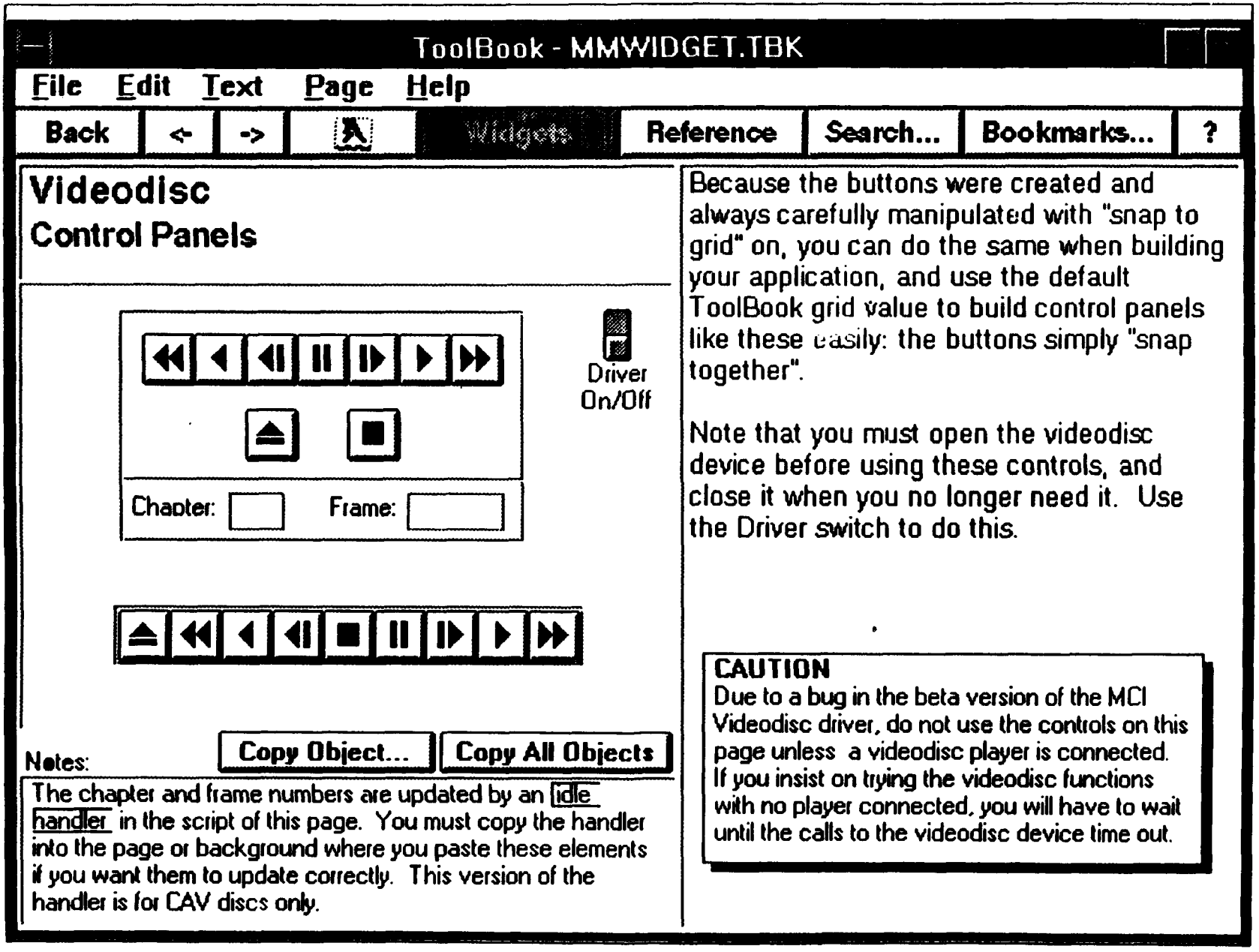

Figure 3.3 Multimedia Video Disk Control Panel

(identified in *ToolBook* scripting language as a "buttondown" message handler). In this respect, *ToolBook* is a true hypermedia-based application builder: there exists a one-to-one mapping between the graphics objects presented on the screen and the script programs which handle their internally represented attributes, behaviors, and links to other data structures/objects. These scripts can be copied for use in user-developed applications. For example, the video sequence player page (Figure 3.2) contains scripts that handle the positioning of the video window on the screen. The videodisc controls panels page (Figure 3.3) contains scripts associated with each button to handle actions to control a videodisc player: rewind, play, fast-forward, pause, etc. The "GERMAN VIDEODISK" has implemented customized versions of these buttons and associated script programs.

There exist three lesson viewing modes for the video lesson interface: "student viewing," "instructor," and a *ToolBook* system "author" mode. Each mode affords a user with a different level of video-audio control options. The default mode is "student viewing." Within these modes, there are various screen view options. The video lesson template features are described in the following sections. Appendix B contains excerpts from the programming language code for the "GERMAN VIDEODISK" lesson.

G. VIDEO LESSON TEMPLATE: VIEWING FEATURES

Figures 3.4, 3.5, and 3.6 illustrate three different screen views of the Video Lesson template. In each figure, the panel of buttons that lines the far right of the screen is the General Controls Panel. The buttons that it displays are used as described in the following sections (functions of each are described from the button at the top of the screen to bottom).

Figure 3.4 Linguist Workstation German Video Disk

Figure 3.5 Linguist Workstation German Video Disk in English

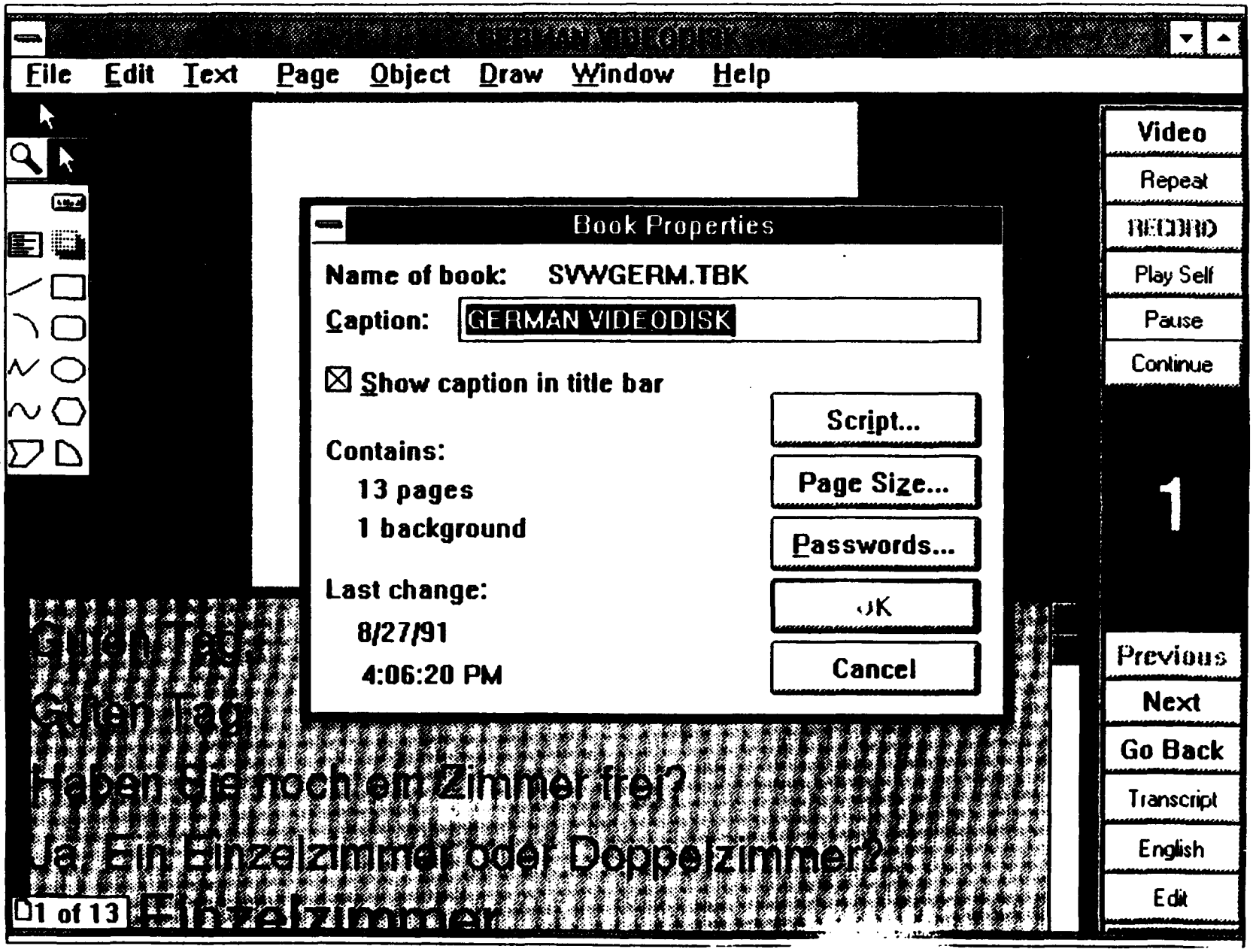

Figure 3.6 Linguist Workstation German Video Disk Authoring Mode

"Video" button: initializes and starts the video from scene "1." When activated, an animated graphic--a black circle spinning over the button--provides feedback to user that the video has started running.

"Repeat" button: plays an instructor-created audio recording whose sound file is associated with the dialogue displayed in the video lesson's transcript window. This sound file is a "permanent" file that is saved from one tutorial session to another.

"RECORD" button: records audio. Activation of this button records voice input via a hand-held microphone. When activated, the General Controls Panel scene number field (center of panel) is highlighted in red, with an animated circle object indicating to the user that recording is taking place. Recording is stopped by toggling this button off.

When in "student viewing" mode, a temporary sound file that is associated with the scene being played on that page of the lesson will be created. With this button, a student can practice dialogue and play back his recordings for pronunciation/inflection comparison with the instructor's recording. This student sound file is not saved from one tutorial session to the next.

When in "instructor" mode, a sound file is created that is associated with the scene being played on that page of the lesson. This sound file is commonly known as the "native speaker" or "target language" recording. It is created as a "permanent" file that is saved from one tutorial session to another and played back via the "Repeat" button. This button also allows an LI to edit the existing sound file.

"Play Self" button: handles playback of student recordings. A student can press this button, hear his own practice recording, then playback the "native speaker" recording (by activating the "Repeat" button) navigate through the pre-sequenced video

("previous," next," "go back"), and, when in "instructor" mode, edit sequences of video for lesson production.

"Pause" and "Continue" buttons: these buttons control pause and continuation of audio playback.

"Scene number" field: displays the video scene being viewed. The transcript dialogue is associated with this scene number of the video lesson page. As noted above, during audio recording, this field turns red, with the scene number replaced by an animated red ball graphic, to provide feedback to the user that recording is taking place.

"Previous" button: plays previous scene.

"Next" button: plays following scenes--each button press takes the sequence forward one scene.

"Go Back" button: replays previous scenes--each button press takes the sequence back one scene.

"Transcript" button: when toggled, this button displays or hides a transcript dialogue field below the video window. Figure 3.4 shows the screen view with the transcript dialogue field toggled on. The transcript shown is in the target language font (in Figure 3.4, the language is German, but the transcript could be in a cyrillic font for Russian, kanji for Japanese. etc.). Its dialogue is associated with the scene being played on the video lesson page being shown. Scroll bars are provided on the far right of the transcript dialoguefield.

When in "instructor" mode, this transcript can be edited as desired by an LI. Development plans include adding hotspot fields for key vocabulary words within the transcript dialogue field. When these hotspots are activated, a text and/or graphic window will be overlayed on the screen showing a text and/or pictorial vocabulary entry in the target language dictionary. These

vocabulary entries will be editable when in "instructor" mode; "read-only" when in "student viewing" mode.

"English" button: when toggled, this button displays or hides an English translation of the transcript dialogue field. When in "instructor" mode, this transcript can be edited as desired by an LI. Figure 3.5 displays the English dialogue field associated with the transcript dialogueshown in Figure 3.4.

Development plans for this button include artificial intelligence application of automatic language transcription between the transcript dialogue field and the English dialogue field. In short, when an LI creates a dialogue for either the target language or English, a dialogue text for the other language transcript field will be automatically generated.

"Edit" button: the features of this button are only activated when in "instructor" mode, and are described in the following paragraphs.

"Controls" button: when toggled, this button displays or hides the the block of control buttons,

[<Scan | Reverse | Pause | Forward | Scan>]

located at the bottom of the video lesson interface.

Both Figures 3.4 and 3.5 show the "GERMAN VIDEODISK" in "instructor" mode, with the video "Transcript" button toggled "on." "Instructor" mode is only available for use by LIs--access is currently controlled via password verification (note: development plans include providing LIs automatic editing powers based on their login identification, i.e., if the current user is listed in the "LI with edit power privileges" password table, a variable enabling the activation of the "Edit" control button is switched "on;" else, the user must be a student and the "Edit" button will not respond to a button press). When the "Edit" button is

activated via mouse button, a production panel window is displayed for use in developing lessons from the video sequences stored on the videodisc. This video production window is located between the video display window and the General Controls Panel. (Figure 3.4, 3.5). Its features include:

"Previous" button: plays previous scene; track counters correspond to position of video sequence on disc--scene numbers are displayed in the General Controls Panel to the right.

"New" button: creates and saves a new scene as indexed in the counter fields.

"Go Back" button: replays previous scenes--each button press takes the sequence back one scene.

Counter fields: from top to bottom, these fields indicate: editor's current position on video, scene start, and scene end marks for the scene that is being played, or has been created previously. The current position field (top) will not necessarily match the scene start/end fields if the video is in continuous play mode, which is activated by the block of control buttons,

[<Scan | Reverse | Pause | Forward | Scan>]

located at the bottom of the video lesson interface.

"Play Scene" button: plays the scene corresponding to the sequence range displayed in the start/stop counter fields.

"Mark Start" button: marks a scene start as indicated by the current scene counter number. For example, in Figure 3.4, the counter shows "02215" is the frame currently being played. The scene already saved as scene "1" falls between the range "13402" to "14938." If an LI were to edit the start from "02215," the start counter field would be saved as "02215," and scene "1" would be reassociated with a play sequence from "02215" to "14938."

"Mark End" button: marks a scene end as indicated by the current scene counter number. The same procedure noted for "Mark Start" applies.

The "Edit" production panel can be hidden from screen view by toggling the "Edit" button in the General Controls Panel. off. The block of control buttons, [<Scan | Reverse | Pause | Forward | Scan>] located at the bottom of the video lesson interface control the continuous play, scan forward/back, rewind, pause, and forward options of video playback. These buttons are needed when by LIs who are editing lessons as well as by linguists who prefer to watch the videodisc scenes as a movie rather than as dialogue and vocabulary-based study segments.

Figure 3.6 shows the "GERMAN VIDEODISK" in *ToolBook's* "author" mode; only system managers and interface module developers are authorized access to this mode. The menubar shown beneath the "GERMAN VIDEODISK" title bar is a ToolBook system feature that provides facilities for creating and editing the scripts that contain the message handler programs associated with the data objects within a ToolBook book. The "Book Properties" window overlayed on the "GERMAN VIDEODISK" page is also a *ToolBook*-specific feature. It displays the name of the book program currently being developed (in this case, "SVWGERM.TBK") as well as the title bar caption ("GERMAN VIDEODISK"). Other information includes the number of objects and pages associated with the book and buttons that will call other features--the book script, the page size options, any passwords associated with the display of scripts associated with the book, and window acceptance ("OK") and dismissal ("Cancel") buttons. The tool palette shown in the upper left corner of Figure 3.6 shows the tools available to a

ToolBook author for object creation/editing. The small field in the lower left corner shows the author "where he is" in the development of the current book, using page numbers. In Figure 3.6, the author is working on page "1 of 13." These system features are themselves examples of hypermedia: buttons and screen objects whose behaviors are activated/deactivated based on user control and selection.

Each video lesson produced by an LI, whether presenting course materials for subsequent German lessons or in different target languages, uses the same interface (and consequently, the same *ToolBook* book template) for presentation. The "KOREAN VIDEODISK," for example, uses the same interface template--the video frame sequences, the title, the transcripts, and associated audio recordings differ in accordance with the design developed by the LI and the videodisc being used. These consistent screen conventions facilitate rapid courseware development for all languages.

VI. CONCLUSIONS

A. ACCOMPLISHMENTS, CURRENT LIMITATIONS AND FUTURE RESEARCH

The major problems associated with the Linguist Workstation research conducted at DLI can be attributed to the beta release software available for use in system design and development. In addition to software "bugs" encountered in any software development process, the untested, undocumented errors embedded within the beta release Multimedia Extensions to Microsoft Windows version 1.0 [Mi91] contributed to many delays and requirements for recoding. With aquisition of the first commercially released version projected for October 1991 [Mi91], many of these system errors will have been corrected and/or documented, and future Workstation development processes should proceed with minimal--or at least, minimized--delays.

Another problem encountered was a result of an inconsistent hardware acquisition process. In the early stages of phase one of the BMLC Project, certain hardware components were recommended, procured and received. As acquisition proceeded, new technological advances and decreasing prices in industry offered new options to project management, and procurement orders changed to reflect these new options. Consequently, some prototype workstations were installed with video and audio components that, coupled with system software bugs, proved incompatible with software and hardware configurations installed on other workstations. A "standard" Workstation model has yet to be developed and estabished. However, as industry standards for

multimedia design and development are established and multimedia software libraries are codified, these incompatibility problems should be minimized.

Within the past year and through the next phase (1992-1993) of Linguist Workstation development, an object-oriented database has been/is being designed for use in linking vocabulary, grammar, and cultural references with video, graphics, and primary source materials. Existing systems that are being used as protoype examples are Harvard University's Perseus Project and Brown University's InterMedia System, among others. Project managers, interface module developers, and LIs alike envision a hypermedia-linked research and publication environment rich in exploration trails for use in both foreign language study and courseware development.

Opportunities for future research include voice recognition and synthesis, automatic natural language translation generation, and expert-system-based development of student profile modules. It is conceivable that hypermedia concepts can be applied to link these technologies and research findings with present system capabilities in an over-arching, comprehensive interdisciplinary integration.

B. A CASE FOR HYPERMEDIA LEARNING ENVIRONMENTS

As stated at the outset of this thesis, a need exists to expand present "page-turner" computer-based instruction applications into comprehensive instructional environments--environments that facilitate "student-centered" learning as well as augment the power of a human instructor. An effective ITS should motivate students by providing enriching environments and creating new microworlds for instruction [Ps91]. Hypermedia offers certain cognitive and learning advantages over other methods of subject matter presentation in its representation-equals-

presentation design. The multi-modal presentation capabilities implicit in hypermedia appear to help improve recall for presented information, reduce short-term memory loads during problem-solving, provide information about the state of a problem solution, help users organize their knowledge about a problem, and increase motivation or user satisfaction [Ca91].

This thesis proposed that hypermedia, along with new directions in cognitive theory research, offers an alternative methodology for exploiting capabilities of new multimedia computer-based resources. A key feature of a hypermedia approach is that a developer is able to design different presentations of the same information customized to the requirements of different learning tasks. From a cognitive science perspective, the utility/effectiveness in presenting concepts in various forms enriches the environment and opportunities for learning without overloading by allowing students to substitute quick perceptual inferences for more demanding logical inferences [Ca91].

The applicability of hypermedia concepts was reviewed and applied with a prototype Linguist Workstation being developed by the Basic Military Language Course (BMLC) Project at the Defense Language Institute (DLI), Presidio of Monterey. The military rationale for this project is based on the portability and exportability of CBT. The military objective is standardized training, which will help to obviate the recurring need for re-training newly arriving personnel. Because motivation is increased with hypermedia's emphasis on user-centered application environments (user-control options facilitate this response), potential exists to improve retention of personnel. The pedagogical rationale is based on research regarding the development of individualized training environments, which could provide empirical data for use in supporting/disproving emerging

cognitive theory hypotheses. Research findings maintain that a hypermedia-based tutorial module can not only provide an interactive learning environment for students but also be an aid to instructors in curricula design.

APPENDIX A

```
;COMMANDPOST.CPM
;;the command post program that is run as a Microsoft Windows shell
;;This program calls the command post program, "Linguist.cpm" which
;;generates the Linguist Workstation Interface

#NextFile linguist.cpm
crlf=strcat(num2char(13),num2char(10))
tab=num2char(9)
ThisWin = WinGetActive()
        errormode(@off)
        dirchange("c:\batch")
        run("macro.exe","")
        winhide("PubTech Macro")
        name="sysop"
        pswd="installation"
        syspswd="installation"
        lang="General"
        loggeddrive="c:\"
        dirchange(loggeddrive)
        dirchange("users")
        dirmake(name)
        a=strcat("The Linguist's Workstation - ",lang)
        errormode(@off)
        wintitle(thiswin,a)
        WinPlace (0,-10,1000,87,"The Linguist's Workstation")
        winactivate("The")

;LINGUIST.CPM

        run("intermis.exe","")
&Applications
 &Al Kaatib International
        if name=="" then display (5,"Who are you?","Please logon by pressing the letter
        ""L"".")
        if name=="" then exit
        errormode(@off)
        dirchange("c:\aki")
        terminate(winactivate("Al Kaatib"),"","")
        errormode(@off)
        a=askline("Al Kaatib","Have you attached the key to the printer port?","No")
        if a=="Yes" then runzoom("aki.exe","")
 Calendar
        if name=="" then display (5,"Who are You?","Please Logon by pressing the
        letter ""L"".")
        if name=="" then exit
        errormode(@off)
```

```
      a=strcat("1 Team - ",name)
      terminate(winactivate(a),"","")
      errormode(@off)
      dirchange("c:\team")
      a=strcat("c:\users\",name,"\",name,".pak")
      run("team.exe",a)
      exit
&Excel
      if name=="" then display (5,"Who are you?","Please logon by
      pressing the letter ""L"".")
      if name=="" then exit
      errormode(@off)
      dirchange("c:\excel")
      terminate(winactivate("Microsoft  Excel"),"","")
      errormode(@off)
      run("excel.exe","")
&Flowcharter
      if name=="" then display (5,"Who are you?","Please logon by
      pressing the letter ""L"".")
      if name=="" then exit
      errormode(@off)
      dirchange("c:\abc")
      terminate(winactivate("ABC"),"","")
      errormode(@cancel)
      run("abc.exe","")
Microsoft &Database
      if name=="" then display (5,"Who are you?","Please logon by
      pressing the letter ""L"".")
      if name=="" then exit
      errormode(@off)
      dirchange("c:\sb4")
      terminate(winactivate("Microsoft  Database"),"","")
      errormode(@cancel)
      runzoom("msdb.exe","")
PageMaker
      if name=="" then display (5,"Who are you?","Please logon by
      pressing the letter ""L"".")
      if name=="" then exit
      errormode(@off)
      dirchange("c:\PM4")
      run("PM4.exe","")
Pa&intb&rush
      if name=="" then display (5,"Who are you?","Please logon by
      pressing the letter ""L"".")
      if name=="" then exit
      errormode(@off)
      terminate(winactivate("Paint"),"","")
      errormode(@cancel)
      runzoom("pbrush.exe","")
&Powerpoint
      if name=="" then display (5,"Who are you?","Please logon by
```

```
      pressing the letter ""L"".")
      if name=="" then exit
      errormode(@off)
      terminate(winactivate("PowerPoint"),"","")
      dirchange("c:\powerpt")
      errormode(@cancel)
      runzoom("powerpwm.exe","")
Pro&ject
      if name=="" then display (5,"Who are you?","Please logon by
      pressing the letter ""L"".")
      if name=="" then exit
      errormode(@off)
      terminate(winactivate("Microsoft  Project"),"","")
      errormode(@cancel)
      dirchange("c:\project")
      runzoom("winproj.exe","")
&ToolBook
      if name=="" then display (5,"Who are you?","Please logon by
      pressing the letter ""L"".")
      if name=="" then exit
      errormode(@off)
      dirchange("c:\lessons")
      dirchange(lang)
      terminate(winactivate("Toolbook"),"","")
      a=diritemize("*.*")
      a=strcat(".. _New ",a)
      a=itemselect("Select a Directory",a," ")
      if a=="_New" then goto new
      dirchange(a)
      a=fileitemize("*.tbk")
      a=itemselect("Select a book",a," ")
      run("toolbook.exe",a)
      exit
      :new
      a=askline("New Directory","Enter the name of the new directory","")
      terminate(a=="","Create Error","Cannot create directory with null name")
      DirMake(a)
      dirchange(a)
      run("c:\toolbook\toolbook.exe","")
Ventura Publisher
      if name=="" then display (5,"Who are you?","Please logon by
      pressing the letter ""L"".")
      if name=="" then exit
      errormode(@off)
      dirchange("c:\ventura")
      run("VPWIN.exe","")
WinComm
      if name=="" then display (5,"Who are you?","Please logon by
      pressing the letter ""L"".")
      if name=="" then exit
      errormode(@off)
```

```
        terminate(winactivate("WinComm"),"","")
        errormode(@cancel)
        dirchange("c:\wincomm")
        run("wincomm.exe","")
WinWord  -  A: Drive
        loggeddrive="a:\"
        errormode(@off)
        dirchange("A:")
        a=fileitemize("*.doc")
        a=strcat(".. _New ",a)
        a=itemselect("Select a file",a," ")
        if a=="_New" then goto new
        run(a,"")
        exit
        :new
        run("Winword.exe","")
&WinWord  -  C: Drive\ W
        if name=="" then display (5,"Who are ycu?","Please logon by
        pressing the letter ""L"".")
        if name=="" then exit
        errormode(@off)
        terminate(winactivate("Microsoft  Word"),"","")
        dirchange("c:\lessons")
        dirchange(lang)
        a=diritemize("*.*")
        a=strcat(".. _New ",a)
        a=itemselect("Select a Dir  tory",a," ")
        if a=="_New" then goto new
        dirchange(a)
        a=fileitemize("*.doc")
        a=strcat(".. _New ",a)
        a=itemselect("Select a document",a," ")
        if a=="_New" then goto newdoc
        run(a,"")
        exit
        :new
        a=askline("New Directory","Enter the name of the new directory","")
        terminate(a=="","Create Error","Cannot create directory with null name")
        errormode(@off)
        DirMake(a)
        dirchange(a)
        runzoom("winword.exe","")
        exit
        :newdoc
        runzoom("winword.exe","")
        exit
        Multimedia Tools
        run("","")
Cloc&k
        if name=="" then display (5,"Who are you?","Please logon by
        pressing the letter ""L"".")
```

```
        if name=="" then exit
        errormode(@off)
        terminate(winactivate("Clock"),"","")
        errormode(@cancel)
        run("mmclock.exe","")
&Midi Player
        if name=="" then display (5,"Who are you?","Please logon by
        pressing the letter ""L"".")
        if name=="" then exit
        errormode(@off)
        terminate(winactivate("Midi"),"","")
        errormode(@cancel)
        dirchange("c:\midi")
        run("midiplay.tbk","")
&Mixer - Audio \X
        if name=="" then display (5,"Who are you?","Please logon by
        pressing the letter ""L"".")
        if name=="" then exit
        errormode(@off)
        terminate(winactivate("Mix"),"","")
        errormode(@cancel)
        dirchange("c:\windows")
        run("mix.exe","")
&Multimedia Player
        if name=="" then display (5,"Who are you?","Please logon by
        pressing the letter ""L"".")
        if name=="" then exit
        errormode(@off)
        dirchange("c:\sb4")
        terminate(winactivate("Multimedia"),"","")
        errormode(@cancel)
        run("mplayer.exe","")
&Multimedia Widgets
        if name=="" then display (5,"Who are you?","Please logon by
        pressing the letter ""L"".")
        if name=="" then exit
        errormode(@off)
        terminate(winactivate("Multimedia"),"","")
        errormode(@cancel)
        dirchange("c:\toolbook")
        run("toolbook.exe","mmwidget.tbk")
Screen Saver
        errormode(@off)
        run("intermis.exe","")
Wa&ve Editor \ V
        if name=="" then display (5,"Who are you?","Please logon by
        pressing the letter ""L"".")
        if name=="" then exit
        errormode(@off)
        terminate(winactivate("Wave"),"","")
        errormode(@cancel)
```

```
      dirchange("c:\mdktools")
      run("waveedit.exe","")
      Utilities
      dirchange("")
&Calculator \=
      if name=="" then display (5,"Who are you?","Please logon by
      pressing the letter ""L"".")
      if name=="" then exit
      errormode(@off)
      terminate(winactivate("Calcul"),"","")
      errormode(@cancel)
      run("calc.exe","")
Clip&board
      if name=="" then display (5,"Who are you?","Please logon by
      pressing the letter ""L"".")
      if name=="" then exit
      errormode(@off)
      terminate(winactivate("Clip"),"","")
      errormode(@cancel)
      run("clipbrd.exe","")
&Control Panel \ ^C
      if name=="" then display (5,"Who are you?","Please logon by
      pressing the letter ""L"".")
      if name=="" then exit
      if pswd==syspswd then goto cpl
      oneshot=askline("Password","What is the System Manager   Password?","")
      if oneshot==syspswd then goto cpl
      exit
      :cpl
      errormode(@off)
      terminate(winactivate("Control"),"","")  ;Already  Running
      errormode(@cancel)
      run("cpl.exe","")
Copy... \ ^{INSERT}
      r=OtherDir()
      a=strsub(";",1,IsKeyDown(@SHIFT))
      s=strcat(DirItemize("")," ",FileItemize(""))
      terminate(strlen(s)==1,"Copy Error","No files selected")
      execute %a% r=askline("Copy",StrCat(s,crlf,crlf,"to"),r)
      terminate(r=="","Copy Error","Cannot copy to null file name")
      q=strindex(r,"\",0,@FWDSCAN) ; Directory Name in there??
      FileCopy(s,r,@TRUE)
      OtherUpdate()
      q=strsub(";",1,q!=0)
      execute %q% SetDisplay("","","")
      drop(a,r,s,q)
&Delete File.. \ {DELETE}
      if name=="" then display (5,"Who are you?","Please logon by
      pressing the letter ""L"".")
      if name=="" then exit
      if pswd==sys swd then goto delete
```

```
        syspswd=askline("Password","What is the System Manager Password?","")
        if syspswd=="installation" then goto delete
        exit
        :delete
        f=FileItemize("")
        g=DirItemize("")
        c=strsub( ";", 1, (strlen(g)==0) )
        b=0
        execute %c% b=AskYesNo("!!! Warning !!! ",strcat("Delete files
        from these directories too?",crlf,g))
        c=strsub(";",1,abs(b-1))
        execute %c% f=strcat(g," ",f)
        terminate(strlen(f)==0,"delete","No files specified")
        terminate(askyesno("Delete",f)==0,"Delete","File(s) not deleted")
        FileDelete(f)
        SetDisplay("","","")
        OtherUpdate()  ; Well if the "other" CmdPost Window points to
        ;the same directory, it *IS* nice...
        drop(f,g,a)
&File Manager     \ ^F
        if name=="" then display (5,"Who are you?","Please logon by
        pressing the letter ""L"".")
        if name=="" then exit
        if pswd==syspswd then goto filemgr
        syspswd=askline("Password","What is the System Manager Password?","")
        if syspswd=="installation" then goto filemgr
        exit
        :filemgr
        errormode(@off)
        erminate(winactivate("File"),"","")
        errormode(@cancel)
        runzoom("winfile.exe","")
&Help \ H
        dirchange("c:\cp")
        errormode(@off)
        terminate(winactivate("1 Team Notebook - LINGHELP"),"","")
        errormode(@cancel)
        run("c:\team\teamnb.exe","linghelp.nbk")
&Keyboard Map \K
        a=askyesno("KEYBOARD LAYOUT", "Do you want to see non-Roman fonts?")
        if a==1 then goto fonts
        dirchange("c:\windows")
        errormode(@off)
        terminate(winactivate("Key"),"","")
        errormode(@cancel)
        run("Keymap.exe","")
        exit
        :Fonts
        dirchange("c:\AKI")
        errormode(@off)
        terminate(winactivate("Key"),"","")
```

```
      errormode(@cancel)
      run("akikbd.exe","")
&Menu Editor \ M
      if name=="" then display (5,"Who are you?","Please logon by
      pressing the letter ""L"".")
      if name=="" then exit
      if pswd==syspswd then goto system
      oneshot=askline("Password","What is the System Manager Password?","")
      if oneshot==syspswd then goto system
      exit
      :system
      dirchange("c:\cp")
      a=FileItemize("*.CPM")
      a=ItemSelect("Select menu to edit",a," ")
      terminate(a=="","Edit","No file selected")
      ErrorMode(@OFF)
      terminate(WinActivate("Notepad - %a%"),"","") ;Already being     edited
      dot=strindex(a,".",0,@FWDSCAN)
      bak=strcat(strsub(a,1,dot),"BAK")
      FileCopy(a,bak,@FALSE)
      run("notepad.exe",a)
&Notepad \^N
      if name=="" then display (5,"Who are you?","Please logon by
      pressing the letter ""L"".")
      if name=="" then exit
      errormode(@off)
      terminate(winactivate("Notepad"),"","")
      errormode(@cancel)
      Run("notepad.exe",currentfile())
P&rint Manager
      run("printman.exe","")    ; takes care of itself
Pr&ogram Manager
      if name=="" then display (5,"Who are you?","Please logon by
      pressing the letter ""L"".")
      if name=="" then exit
      if pswd==syspswd then goto system
      oneshot=askline("Password","What is the System Manager Password?","")
      if oneshot==syspswd then goto system
      exit
      :system
      errormode(@off)
      terminate(winactivate("Program Manager"),"","") ;Already  Running
      errormode(@cancel)
      run("Progman.exe","")
&System Manager \ S
      if name=="" then display (5,"Who are you?","Please logon by
      pressing the letter ""L"".")
      if name=="" then exit
      if pswd==syspswd then goto system
      oneshot=askline("Password","What is the System Manager Password?","")
      if oneshot==syspswd then goto system
```

```
        exit
        :system
        errormode(@off)
        terminate(winactivate("System"),"","")
        errormode(@cancel)
        run("system.cpm","")
 &System Configuration \ ^S
        if name=="" then display (5,"Who are you?","Please logon by
        pressing the letter ""L"".")
        if name=="" then exit
        if pswd==syspswd then goto system
        oneshot=askline("Password","What is the System Manager   Password?","")
        if oneshot==syspswd then goto system
        exit
        :system
        run("sysedit.exe","")
 &Text Editor \^N
        if name=="" then display (5,"Who are you?","Please logon by
        pressing the letter ""L"".")
        if name=="" then exit
        errormode(@off)
        terminate(winactivate("Pubtech"),"","")
        errormode(@cancel)
        dirchange("c:\pubtech")
        Run("text.exe",currentfile())
_&Logon and off \ L
        errormode(@off)
        WinClose("Sys")
        pswd=""
        name=""
        lang="  YOU NEED TO LOGON!      Please press ""L"""
        errormode(@off)
        a=strcat("The Linguist's Workstation - ",lang)
        wintitle("The Ling",a)
        :logon
        WinPlace (0,0,1000,45,"The Linguist's Workstation")
        name=askline("LOGON,  PLEASE","User, enter your last name:","")
        if name=="" then goto logon
        :continue
        name=strupper(name)
        dirchange("c:\users")
        a=dirchange(name)
        if name=="SYSOP" then goto sysop
        if a==0 then goto verify
        WinPlace (0,-10,1000,87,"The Linguist's Workstation")
        if name!="" then goto select
        lang=""
        pswd=""
        a=strcat("The Linguist's Workstation - ",lang)
        wintitle("The Ling",a)
        exit
```

```
        :verify
        b=askline("NAME","Is your name spelled correctly?",name)
        b=strupper(b)
        if b==name then goto welcome
        name=b
        goto continue
        :welcome
        c=askline("WELCOME NEW USER","Please type your name again
        for verification.","")
        c=strupper(c)
        if c!=name then goto verify
        name=c
        dirmake(name)
        goto continue
        exit
        :sysop
        name="SYSOP"
        pswd=syspswd
        lang="General"
        a=strcat("The Linguist's Workstation - ",lang)
        errormode(@off)
        wintitle("The Ling",a)
        WinPlace (0,-10,1000,87,"The Linguist's Workstation")
        exit
        :select
        dirchange("c:\lessons")
        lang=""
        a=diritemize("*.*")
        lst=strcat("_New ", a)
        lang=ItemSelect("Select a language:",lst," ")
        if lang!="_New" then goto theend
        lang=askline("New Language","Which Language do you want to add?","")
        if lang=="" then go  sellect
        dirmake(lang)
        :theend
        if lang=="" then lang="General"
        a=strcat("The Linguist's Workstation - ",lang)
        wintitle("The Ling",a)
        WinPlace (0,-10,1000,87,"The Linguist's Workstation")
        errormode(@off)
!Games
        dirchange("C:\winshare")
 &Cruel
        if name=="" then display (5,"Who are you?","Please logon by
        pressing the letter ""L"".")
        if name=="" then exit
        dirchange("c:\winshare")
        run("c:\winshare\cruel.exe","")
&Golf
        if name=="" then display (5,"Who are you?","Please logon by
        pressing the letter ""L"".")
```

```
        if name=="" then exit
        dirchange("c:\winshare")
        run("c:\winshare\golf.exe","")
 Klot&z (Tetris)
        if name=="" then display (5,"Who are you?","Please logon by pressing the letter
""L"".")
        if name=="" then exit
        dirchange("c:\winshare")
        run("klotz.exe","")
WinPlace(333,0,666,1000,"K")
 L&unar Lander
        if name=="" then display (5,"Who are you?","Please logon by
        pressing the letter ""L"".")
        if name=="" then exit
        dirchange("c:\winshare")
        runzoom("lander.exe","")
 &Pegged
        if name=="" then display (5,"Who are you?","Please logon by
        pressing the letter ""L"".")
        if name=="" then exit
        dirchange("c:\winshare")
        run("c:\winshare\pegged.exe","")
 &Reversi
        if name=="" then display (5,"Who are you?","Please logon by
        pressing the letter ""L"".")
        if name=="" then exit
        run("reversi.exe","")
 &Solitaire
        if name=="" then display (5,"Who are you?","Please logon by
        pressing the letter ""L"".")
        if name=="" then exit
        runzoom("sol.exe","")
 &Taipei
        if name=="" then display (5,"Who are you?","Please logon by
        pressing the letter ""L"".")
        if name=="" then exit
        dirchange("c:\winshare")
        run("c:\winshare\tp.exe","")
 &Tetris
        if name=="" then display (5,"Who are you?","Please logon by
        pressing the letter ""L"".")
        if name=="" then exit
        dirchange("c:\winshare")
        run("c:\winshare\tetris.exe","")
 &Tic 3D
        if name=="" then display (5,"Who are you?","Please logon by
        pressing the letter ""L"".")
        if name=="" then exit
        dirchange("c:\winshare")
        run("c:\winshare\tic.exe","")
 &Minefield (Battleship)
```

```
        if name=="" then display (5,"Who are you?","Please logon by
        pressing the letter ""L"".")
        if name=="" then exit
        dirchange("c:\winshare")
        run("c:\winshare\winmine.exe","")
&Lessons
 &BMLC Textbook \W
        if name=="" then display (5,"Who are you?","Please logon by
        pressing the letter ""L"".")
        if name=="" then exit
        errormode(@off)
        terminate(winactivate("Microsoft  Word"),"","")
        dirchange("c:\lessons")
        dirchange(lang)
        a=diritemize("*.*")
        a=strcat(".. _New ",a)
        a=itemselect("Select a Directory",a," ")
        if a=="_New" then goto new
        dirchange(a)
        a=fileitemize("*.doc")
        a=strcat(".. _New ",a)
        a=itemselect("Select a document",a," ")
        if a=="_New" then goto newdoc
        run(a,"")
        exit
        :new
        a=askline("New Directory","Enter the name of the new
        directory","")
        terminate(a=="","Create Error","Cannot create directory with
        null name")
        errormode(@off)
        DirMake(a)
        dirchange(a)
        runzoom("winword.exe","")
        exit
        :newdoc
        runzoom("winword.exe","")
        exit
 &BMLC CBT Developmental Lessons \T
        if name=="" then display (5,"Who are you?","Please logon by
        pressing the letter ""L"".")
        if name=="" then exit
        errormode(@off)
        dirchange("c:\lessons")
        dirmake(lang)
        dirchange(lang)
        terminate(winactivate("Toolbook"),"","")
        a=diritemize("*.*")
        a=strcat(".. _New ",a)
        a=itemselect("Select a Directory",a," ")
        if a=="_New" then goto new
```

```
        dirchange(a)
        a=fileitemize("*.tbk")
        a=strcat(".. _New ",a)
        a=itemselect("Select a book",a," ")
        if a=="_New" then goto newbook
        run(a,"")
        exit
        :new
        a=askline("New Directory","Enter the name of the new
        directory","")
        terminate(a=="","Create Error","Cannot create directory with
        null name")
        DirMake(a)
        dirchange(a)
        run("c:\toolbook\toolbook.exe","")
        exit
        :newbook
        run("c:\toolbook\toolbook.exe","")
&Homework
        errormode(@off)
        dirchange("c:\lessons")
        dirchange(lang)
        terminate(winactivate("Homework"),"","")
        errormode(@off)
        a=FileItemize("*.hom")
        a=ItemSelect("Select a lesson:",a," ")
        errormode(@off)
        if a=="" then exit
        run("c:\toolbook\toolbook.exe",a)
        exit
&Change Language
        if name=="" then display (5,"Who are you?","Please logon by
        pressing the letter ""L"".")
        if name=="" then exit
        errormode(@off)
        dirchange("c:\lessons")
        a=diritemize("*.*")
        lst=strcat("_New ", a)
        lang=ItemSelect("Select a language:",lst," ")
        if lang<>"_New" then goto oldlang
        lang=askline("New Language","Which Language do you want to
        add?","")
        if lang!="" then dirmake(lang)
        :oldlang
        if lang=="" then lang="English"
        a=strcat("The Linguist's Workstation - ",lang)
        wintitle("The L",a)
        errormode(@off)
_ Browse
        dirchange("c:\lessons")
&Alphabet \l
```

```
      if name=="" then display (5,"Who are you?","Please logon by
      pressing the letter ""L"".")
      if name=="" then exit
      errormode(@off)
      dirchange("c:\lessons")
      dirchange(lang)
      terminate(winactivate("Alphabet"),"","")
      errormode(@cancel)
      a=FileItemize("*.abc")
      a=ItemSelect("Select an alphabet lesson:",a," ")
      if a=="" then exit
      errormode(@cancel)
      run("c:\toolbook\toolbook.exe",a)
&Vocabulary and Spelling \2
      if name=="" then display (5,"Who are you?","Please logon by
      pressing the letter ""L"".")
      if name=="" then exit
      errormode(@off)
      dirchange("c:\lessons")
      dirchange(lang)
      terminate(winactivate("Vocabulary"),"","")
      errormode(@cancel)
      a=ItemSelect("Select a vocabulary lesson:",a," ")
      if a=="" then exit
      errormode(@cancel)
      run("c:\toolbook\toolbook.exe",a)
&Grammar Practice \3
      if name=="" then display (5,"Who are you?","Please logon by
      pressing the letter ""L"".")
      if name=="" then exit
      errormode(@off)
      dirchange("c:\lessons")
      dirchange(lang)
      terminate(winactivate("Grammar"),"","")
      errormode(@cancel)
      a=FileItemize("*.gra")
      a=ItemSelect("Select a grammar lesson:",a," ")
      if a=="" then exit
      errormode(@cancel)
      run("c:\toolbook\toolbook.exe",a)
&Phrasebooks \4
      if name=="" then display (5,"Who are you?","Please logon by
      pressing the letter ""L"".")
      if name=="" then exit
      errormode(@off)
      dirchange("c:\lessons")
      dirchange(lang)
      terminate(winactivate("Phrase"),"","")
      errormode(@cancel)
      a=FileItemize("*.Phr")
      a=ItemSelect("Select a phrasebook lesson:",a," ")
```

```
        if a==""" then exit
        errormode(@cancel)
        run("c:\toolbook\toolbook.exe",a)
Foreign &Broadcasts \5
        if name==""" then display (5,"Who are you?","Please logon by
        pressing the letter ""L"".")
        if name==""" then exit
        errormode(@off)
        dirchange("c:\lessons")
        dirchange(lang)
        terminate(winactivate("Broad"),"","")
        errormode(@cancel)
        a=FileItemize("*.bro")
        a=ItemSelect("Select a foreign broadcast lesson:",a," ")
        if a==""" then exit
        errormode(@cancel)
        run("c:\toolbook\toolbook.exe",a)
&Composition Practice \6
        if name==""" then display (5,"Who are you?","Please logon by
        pressing the letter ""L"".")
        if name==""" then exit
        errormode(@off)
        dirchange("c:\lessons")
        dirchange(lang)
        terminate(winactivate("Composit"),"","")
        errormode(@off)
        ;a=dirItemize("*.*")
        ;a=ItemSelect("Select a subject:",a," ")
        ;dirchange(a)
        a=FileItemize("*.com")
        a=ItemSelect("Select a composition lesson:",a," ")
        if a==""" then exit
        errormode(@cancel)
        run("c:\toolbook\toolbook.exe",a)
WINCALIS \7
        if name==""" then display (5,"Who are you?","Please logon by
        pressing the letter ""L"".")
        if name==""" then exit
        errormode(@cancel)
        dirchange("c:\lessons")
        dirchange(lang)
        dirchange("c:\WINCALIS")
        ;a=dirItemize("*.*")
        ;a=ItemSelect("Select a subject:",a," ")
        ;dirchange(a)
        ;if a==""" then exit
        errormode(@cancel)
        run("wincalis.exe","")
&References
 &Bibliography
        if name==""" then display (5,"Who are you?","Please logon by
```

```
      pressing the letter ""L"".")
      if name=="" then exit
      errormode(@off)
      dirchange("c:\lessons")
      dirchange(lang)
      terminate(winactivate("Bibliography"),"","")
      display(5,"Bibliography","This reference work has not yet been
      completed.")
      exit
      errormode(@cancel)
      run("bibliog.tbk","")
&Dictionary
      if name=="" then display (5,"Who are you?","Please logon by
      pressing the letter ""L"".")
      if name=="" then exit
      errormode(@off)
      dirchange("c:\lessons")
      dirchange(lang)
      terminate(winactivate("Dictionary"),"","")
      display(5,"Dictionary","This reference work has not yet been
      completed.")
      exit
      errormode(@cancel)
      run("diction.tbk","")
&Encyclopedia
      if name=="" then display (5,"Who are you?","Please logon by
      pressing the letter ""L"".")
      if name=="" then exit
      errormode(@off)
      dirchange("c:\lessons")
      dirchange(lang)
      terminate(winactivate("Encyclopedia"),"","")
      display(5,"Multilingual Encyclopedia","This reference work has
      not yet been completed.")
      exit
      errormode(@cancel)
      run("encyclo.tbk","")
&Grammar
      if name=="" then display (5,"Who are you?","Please logon by
      pressing the letter ""L"".")
      if name=="" then exit
      errormode(@off)
      dirchange("c:\lessons")
      dirchange(lang)
      terminate(winactivate("Grammar"),"","")
      display(5,"Grammar","This reference work has not yet been
      completed.")
      exit
      errormode(@cancel)
      run("grammar.tbk","")
&Quotations
```

```
        if name=="" then display (5,"Who are you?","Please logon by
        pressing the letter ""L"".")
        if name=="" then exit
        errormode(@off)
        dirchange("c:\lessons")
        dirchange(lang)
        terminate(winactivate("Quotations"),"","")
        display(5,"Multilingual Quotations","This reference work has
        not yet been completed.")
        exit
        errormode(@cancel)
        run("Quote.tbk","")
&Tutorials
 &Linguist Workstation Help \ H
        dirchange("c:\cp")
        errormode(@off)
        terminate(winactivate("1 Team Notebook - LINGHELP"),"","")
        errormode(@cancel)
        run("c:\team\teamnb.exe","linghelp.nbk")
 Tool&Book Aids
        if name=="" then display (5,"Who are you?","Please logon by
        pressing the letter ""L"".")
        if name=="" then exit
        errormode(@off)
        terminate(winactivate("Toolbook"),"","")
        errormode(@cancel)
        dirchange("c:\toolbook")
        a=fileitemize("*.tbk")
        a=itemselect("Select a book",a," ")
        run(a,"")
 Tool&Book Tutorial
        if name=="" then display (5,"Who are you?","Please logon by
        pressing the letter ""L"".")
        if name=="" then exit
        dirchange("c:\toolbook")
        errormode(@off)
        terminate(winactivate("Quicktour"),"","")
        errormode(@cancel)
        run("c:\toolbook\quiktour.tbk","")
 Windows &Help Files
        if name=="" then display (5,"Who are you?","Please logon by
        pressing the letter ""L"".")
        if name=="" then exit
        errormode(@off)
        dirchange("c:\windows")
        a=FileItemize("*.hlp")
        a=ItemSelect("Select a Help file:",a," ")
        errormode(@cancel)
        run("winhelp.exe",a)
 &WinWord Tutorial
        if name=="" then display (5,"Who are you?","Please logon by
```

```
        pressing the letter ""L"".")
        if name=="" then exit
        display("5","Word for Windows Tutorial","Select     ""Tutorial""
        from the  Help menu after Word For Windows loads")
        dirchange("c:\users")
        dirchange(name)
        runzoom("winword.exe","")
&Personal Information Manager
 &Addresses and Phone #'s \ P
        if name=="" then display (5,"Who are You?","Please Logon by
        pressing the letter ""L"".")
        if name=="" then exit
        errormode(@off)
        terminate(winactivate("1 Team Notebook - PHONE"),"","")
        errormode(@cancel)
        a=itemselect("Which Phone Directory?","Personal General","  ")
        if a=="Personal" then goto personal
        dirchange("c:\users")
        run("c:\team\teamnb.exe","Phonegen.nbk")
        WinPlace(218,177,759,893,"1 Team Notebook - PHONEGEN.NBK")
        exit
        :personal
        dirchange("c:\users")
        errormode(@off)
        dirchange(name)
        run("c:\team\teamnb.exe","phone.nbk")
        WinPlace(218,177,759,893,"1 Team Notebook - PHONE.NBK")
 &Appointments \A
        if name=="" then display (5,"Who are You?","Please Logon by
        pressing the letter ""L"".")
        if name=="" then exit
        errormode(@off)
        a=strcat("1 Team - ",name)
        terminate(winactivate(a),"","")
        errormode(@cancel)
        dirchange("c:\team")
        a=strcat("c:\users\",name,"\",name,".pak")
        errormode(@off)
        run("team.exe",a)
        exit
 Do&cuments
        if name=="" then display (5,"Who are You?","Please Logon by
        pressing the letter ""L"".")
        if name=="" then exit
        errormode(@off)
        dirchange(loggeddrive)
        dirchange("users")
        dirchange(name)
        a=diritemize("*.*")
        a=strcat(".. _New ",a)
        a=itemselect("Select a Directory",a,"  ")
```

```
        if a=="_New" then goto new
        dirchange(a)
        a=fileitemize("*.doc")
        a=strcat(".. _New ",a)
        a=itemselect("Select a document",a," ")
        if a=="_New" then goto newdoc
        if a=="" then exit
        run(a,"")
        exit
        :new
        a=askline("New Directory","Enter the name of the new
        directory","")
        terminate(a=="","Create Error","Cannot create directory with
        null name")
        errormode(@off)
        DirMake(a)
        dirchange(a)
        runzoom("winword.exe","")
        exit
        :newdoc
        runzoom("winword.exe","")
        exit
&Databases \ D
        if name=="" then display (5,"Who are You?","Please Logon by
        pressing the letter ""L"".")
        if name=="" then exit
        dirchange(loggeddrive)
        dirchange("users")
        errormode(@off)
        dirchange(name)
        a=diritemize("*.*")
        a=strcat(".. _New ",a)
        a=itemselect("Select a Directory",a," ")
        if a=="_New" then goto new
        errormode(@off)
        dirchange(a)
        a=fileitemize("*.sbf")
        a=strcat("_New ",a)
        a=itemselect("Select a database:",a," ")
        dirchange("c:\sb4")
        if a=="_New" then goto newbase
        run(a,"")
        exit
        :new
        a=askline("New Directory","Enter the name of the new
        d rectory","")
        terminate(a=="","Create Error","Cannot create directory with
        null name")
        errormode(@off)
        DirMake(a)
        dirchange(a)
```

```
      :newbase
      runzoom("msdb.exe","")
&Financial Accounts \ F
      if name=="" then display (5,"Who are You?","Please Logon by
      pressing the letter ""L"".")
      if name=="" then exit
      dirchange("c:\users")
      errormode(@off)
      dirchange(name)
      errormode(@off)
      dirmake("finance")
      terminate(winactivate("MoneyCounts"),"","")
      errormode(@cancel)
      ;run("c:\mc\mc.exe","")
      display(5,"MoneyCounts","This software will be installed if
      required.  Please ask the System Manager")
Le&tters
      if name=="" then display (5,"Who are You?","Please Logon by
      pressing the letter ""L"".")
      if name=="" then exit
      errormode(@off)
      dirchange(loggeddrive)
      dirchange("users")
      dirchange(name)
      dirmake("letters")
      dirchange("letters")
      a=diritemize("*.*")
      a=strcat(".. _New ",a)
      a=itemselect("Select a Directory",a," ")
      if a=="_New" then goto new
      dirchange(a)
      a=fileitemize("*.doc")
      a=strcat(".. _New ",a)
      a=itemselect("Select a letter",a," ")
      if a=="_New" then runzoom("winword.exe","")
      if a=="" then exit
      runzoom(a,"")
      exit
      :new
      a=askline("New Directory","Enter the name of the new
      directory","")
      terminate(a=="","Create Error","Cannot create directory with
      null name")
      DirMake(a)
      runzoom("winword.exe","")
      dirchange(a)
      exit
&Lists
      if name=="" then display (5,"Who are You?","Please Logon by
      pressing the letter ""L"".")
      if name=="" then exit
```

```
        dirchange("c:\users")
        errormode(@off)
        dirchange(name)
        a=fileitemize("*.nbk")
        a=strcat("New ",a)
        a=itemselect("Lists",a," ")
        if a=="" then exit
        errormode(@cancel)
        if a=="_New" then a=askline("New List","Name of New LIst?","")
        run("c:\team\teamnb.exe",a)
&Notes \ N
        if name=="" then display (5,"Who are You?","Please Logon by
        pressing the letter ""L"".")
        if name=="" then exit
        dirchange("c:\users")
        errormode(@off)
        dirchange(name)
        terminate(winactivate("1 Team Notebook - NOTES"),"","")
        errormode(@cancel)
        run("c:\team\teamnb.exe","notes.nbk")
        WinPlace(68,145,920,908,"1 Team Notebook - NOTES.NBK")
 Pre&sentations
        if name=="" then display (5,"Who are You?","Please Logon by
        pressing the letter ""L"".")
        if name=="" then exit
        dirchange("c:\users")
        errormode(@off)
        dirchange(name)
        terminate(winactivate("PowerPoint"),"","")
        a=diritemize("*.*")
        a=strcat(".. _New ",a)
        a=itemselect("Select a Directory",a," ")
        if a=="_New" then goto new
        dirchange(a)
        a=fileitemize("*.ppt")
        a=strcat(".. _New ",a)
        a=itemselect("Select a Presentation",a," ")
        if a=="_New" then goto newbrief
        errormode(@off)
        runzoom (a,"")
        exit
        :newbrief
        runzoom ("c:\powerpt\powerpw.exe","")
        exit
        :new
        a=askline("New Directory","Enter the name of the new
        directory","")
        terminate(a=="","Create Error","Cannot create directory with
        null name")
        DirMake(a)
        dirchange(a)
```

```
      runzoom("c:\powerpt\powerptwm.exe","")
Pro&jects
      if name=="" then display (5,"Who are You?","Please Logon by
      pressing the letter ""L"".")
      if name=="" then exit
      errormode(@off)
      dirchange("c:\users")
      dirchange(name)
      a=diritemize("*.*")
      a=strcat(".. _New ",a)
      a=itemselect("Select a Directory",a," ")
      if a=="_New" then goto new
      dirchange(a)
      a=fileitemize("*.*")
      a=itemselect("Select a file",a," ")
      run(a,"")
      exit
      :new
      a=askline("New Directory","Enter the name of the new
      directory","")
      terminate(a=="","Create Error","Cannot create directory with
      null name")
      DirMake(a)
      dirchange(a)
      runzoom("winproj.exe","")
Software Information
      if name=="" then display (5,"Who are you?","Please logon by
      pressing the letter ""L"".")
      if name=="" then exit
      errormode(@off)
      dirchange("c:\software")
      terminate(winactivate("Software"),"","")
      errormode(@cancel)
      run("software.nbk","")
Spreadsheets
      if name=="" then display (5,"Who are You?","Please Logon by
      pressing the letter ""L"".")
      if name=="" then exit
      errormode(@off)
      dirchange("c:\users")
      dirchange(name)
      a=diritemize("*.*")
      a=strcat(".. _New ",a)
      a=itemselect("Select a Directory",a," ")
      if a=="_New" then goto new
      dirchange(a)
      a=fileitemize("*.xld")
      a=itemselect("Select a file",a," ")
      if a=="" then exit
      run(a,"")
      exit
```

```
        :new
        a=askline("New Directory","Enter the name of the new
        directory","")
        terminate(a=="","Create Error","Cannot create directory with
        null name")
        DirMake(a)
        dirchange(a)
        runzoom("excel.exe","")
_Communications
        dirchange("")
Local master
        if name=="" then display (5,"Who are You?","Please Logon by
        pressing the letter ""L"".")
        if name=="" then exit
        errormode(@off)
        dirchange("c:\wincomm\session")
        run("c:\wincomm\wincomm.exe","local.wsf")
Local slave
        if name=="" then display (5,"Who are You?","Please Logon by
        pressing the letter ""L"".")
        if name=="" then exit
        errormode(@off)
        run("bwinlink.pif","")
Workstation BBS \ B
        if name=="" then display (5,"Who are You?","Please Logon by
        pressing the letter ""L"".")
        if name=="" then exit
        errormode(@off)
        terminate(winactivate("1 Team Notebook - BBS"),"","")
        errormode(@cancel)
        dirchange("c:\users")
        runzoom("c:\team\teamnb.exe","bbs.nbk")
_Compuserve
        if name=="" then display (5,"Who are you?","Please logon by
        pressing the letter ""L"".")
        if name=="" then exit
        errormode(@off)
        terminate(winactivate("WinComm"),"","") ;Already Running
        dirchange("c:\wincomm")
        run("WinComm.exe","compuser.wsf")
LingNet
        if name=="" then display (5,"Who are you?","Please logon by
        pressing the letter ""L"".")
        if name=="" then exit
        errormode(@off)
        terminate(winactivate("WinComm"),"","") ;Already Running
        errormode(@cancel)
        dirchange("c:\wincomm")
        run("WinComm.exe","Lingnet.wsf")
NiteLog
        if name=="" then display (5,"Who are you?","Please logon by
```

```
      pressing the letter ""L"".")
      if name==="" then exit
      errormode(@off)
      terminate(winactivate("WinComm"),"","") ;Already Running
      errormode(@cancel)
      dirchange("c:\wincomm")
      run("WinComm.exe","Nitelog.wsf")
_Capture Files
      if name==="" then display (5,"Who are You?","Please Logon by
      pressing the letter ""L"".")
      if name==="" then exit
      errormode(@off)
      dirchange("c:\email\CAPTURE")
      a=fileitemize("*.*")
      a=itemselect("Select a capture file",a," ")
      if a==="" then exit
      b=askyesno("Text Editor","Do you want to use Notepad instead
      of WinWord?")
      if b==0 then goto ww
      run("notepad.exe",a)
      exit
      :ww
      run("Winword.exe",a)
Shez Archive Manager
      if name==="" then display (5,"Who are You?","Please Logon by
      pressing the letter ""L"".")
      if name==="" then exit
      dirchange("c:\shez")
      run("shez.exe","")
Zip Archive Manager
      if name==="" then display (5,"Who are You?","Please Logon by
      pressing the letter ""L"".")
      if name==="" then exit
      dirchange("c:\shez")
      run("zm.exe","")
```

APPENDIX B

```
--ToolBook script for German Video Book


--handles the size of the video window
to get ResizeVideo VRect
        system s_Video, s_VBounds
        local l_VC, l_Rtn

        set l_Rtn to FALSE;
        if s_Video <> null
                if VRect <> s_VBounds
                        put VRect into s_VBounds
                -- remove comma's from s_VBounds and place in VC
                        set l_VC to null
                        step i from 1 to charcount(s_VBounds)
                                get character i of s_VBounds
                                if chartoansi(it) <> charToAnsi(",")
                                        put it after l_VC
                                else
                                        put ansiToChar(32) after l_VC
                                end
                        end
                        get AVmove(s_Video, word 1 of l_VC,  word 2 of l_VC)
                        get AVsize(s_Video, word 3 of l_VC - word 1 of l_VC, \
                                        word 4 of l_VC - word 2 of l_VC)
                end
                set l_Rtn to TRUE
        end

        return l_Rtn
end ResizeVideo

to handle enterbook
        system s_WavPath, editmode
        system s_Video
        system s_VBounds
        system s_VBoundsDefault
        system s_fFullPage
        system s_fFullScreen
        system s_fDisplayFrame
        system s_fPausePlayer
        system s_ColorKey
        system currentrect
    set currentrect to "Full Page"
        set s_Video to null
        set s_VBounds to null
```

```
        --set s_VBoundsDefault to bounds of VIDRECT
        set s_fFullPage to FALSE
        set s_fFullScreen to FALSE
        set s_DisplayFrame to FALSE
        set s_PausePlayer to FALSE
        set s_ColorKey to 45

--ensure that the system book that handles messages to video board are put
--on system book stack for messages forwarded to it
    if "svwvideo.sbk" is not in sysbooks
            push "c:\toolbook\svwvideo.sbk" onto sysbooks
   end
        send SVWINIT

        linkDLL "TBKWIN.DLL"
            STRING screenFromPage(WORD, STRING, INT, STRING)
        end linkDLL

        put AVvideoWindow("Source1", 115, 55, 300, 225) into s_VIDEO
        --get AVsetDisplay(s_Video, 1, 0)  -- set hue
        --get AVsetDisplay(s_Video, 2, 40) -- set staturation
        --get AVsetDisplay(s_Video, 3, 75) -- set brightness
        --get AVsetDisplay(s_Video, 4, 40) -- set contrast
        --get AVsetDisplay(s_Video, 5, 70) -- set sharpness
        get AVcolorKey(s_Video, 1, s_ColorKey)

    forward
     if "tbkmm.sbk" is not in sysbooks
            push "c:\toolbook\tbkmm.sbk" onto sysbooks
   end
     if "ld-v2200.tbk" is not in sysbooks
          push "c:\toolbook\ld-v2200.tbk" onto sysbooks
   end
     if videodisc "Open", defcomport of this book) < 0
          request "Cannot initialize the videodisc. Video controls will have"\
       && "no effect."
   end
     send reader
     hide scrollbar
     send sizetopage
     set lessoncode of this book to "G0101"
      set s_WavPath to "c:\lessons\german\"
     set editmode to student
end
to handle RIGHTBUTTONUP fLocation, fIsShift, fIsCtrl
    if fIsShift
       if fIsCtrl
           edit script of this book
      else
           edit script of parent of target
       end if --fIsCtrl
```

```
    else
        if fIsCtrl
            edit script of this background
        else
            edit script of target
        end if --fIsCtrl
    end if --fIsShift
end RIGHTBUTTONUP

to handle buttonDoubleClick
    if mb of this book is true
        hide menubar
        set mb of this book to false
    else
        show menubar
        set mb of this book to true
    end
end

to handle leavebook
    system s_Video
    get videodisc("Close")
    restore system
        --send SVWEND s_Video
    forward
end

to get isNumber x
    set suspend to sysSuspend
    set sysSuspend to false
    set sysError to null
    get x + 0
    set sysSuspend to suspend
    return sysError is null
end
```

```
--ToolBook script for the page object of the German Video Book

--the following handler plays the video clip associated with any page that
--a user happens to be viewing/editing
to handle playsegment
    set f1 to text of recordfield "start frame"
    set f2 to text of recordfield "end frame"
    if isnumber(f1) and isnumber(f2)
       get videodisc("play from" && f1 && "to" && f2)
   end
end

--this handler forwards a message to the book level script to set the
--"author" mode on and hide the scrollbar
to handle author
       forward
       hide scrollbar
end


--handler to flip to "next page, unless user is on last page
to handle next
    if this page <> last page of this background
      forward
   end
end


--sets the color of the video window so that video is visible when playing
to handle disable
    set strokecolor of target to 0,50.1875,0
end

--sets video window to black once video is not playing
to handle enable
    set strokecolor of target to black
end


--takes care of housekeeping for each page (i.e., where are we in
--pagenumbers, display this information to user
to handle enterpage
     system s_video_level, s_audio_level, s_OK
    if text of recordfield "scene" <> pageNumber of this page
        set text of recordfield "scene" to pageNumber of this page
   end
    if this page is last page of this background
        send disable to button "Next" of this background
       show button "New" of this background
       send enable to button "New" of this background
       Show button "First" of this background
```

```
    else
        send enable to button "Next" of this background
        hide button "First" of this background
        hide button "New" of this background
    end
     if this page is first page of this background
        send disable to button "Previous" of this background
    else
        send enable to button "Previous" of this background
    end
end


--Takes care of closing down, cleaning up
to handle leavePage
     system s_OK, s_full_flag
    if s_full_flag <> null
        send exitFullScreen
    end
end


--forwards a message that user wants to quit from lesson
to handle leaveBook
     system s_full_flag
    if s_full_flag <> null
        send exitFullScreen
    end
    forward
end


--used when user views transcript of dialogue; large screen must be
--toggled to a small screen that accommodates all
to handle showFullPage
      system s_full_flag, s_norm_b, s_norm_sz, s_old_f
      system s_video_zoomed, s_old_z, s_page_flag, s_old_f, currentrect
     hide rectangle "Inset Rectangle" of this background
     set currentrect to "Full Page"
    send idle
     show recordfield title
    set s_old_f to 0
    if s_page_flag <> null
        set s_page_flag to null
        break showFullPage
    end
     set s_page_flag to 1
     set s_norm_b to bounds of mainWindow
     set s_norm_sz to size of this book
     show rectangle "Full Page" of this background
     set s_video_zoomed to null
```

```
    set s_old_z to 1
    hide scrollbar
end

--the following scripts illustrate some more of ToolBook's scanning and
--review features
to handle showFullScreen
     system s_full_flag, s_norm_b, s_norm_sz, s_video_zoomed, s_old_z
     system s_fullScreen, s_old_f, s_page_flag, currentrect
    hide rectangle "Inset Rectangle" of this background
    set currentrect to "Full Rect"
   send idle
    hide recordfield "title" of this background
       set s_old_f to 0
       set s_page_flag to 0
   if s_full_flag <> null
       set s_full_flag to null
       break showFullScreen
  end
    --set s_fullScreen to 2
    --set syslockscreen to true
   set s_full_flag to 1
   set s_norm_b to bounds of mainWindow
   set s_norm_sz to size of this book
    show rectangle "Full Rect" of this background
   set size of this book to 640* 15, 480 * 15
    set bounds of mainWindow to -3,-22,643,483
    --hide rectangle "zoom target" of this background
    --hide field "zoom prompt" of this background
   set s_video_zoomed to null
   set s_old_z to 1
    hide scrollbar
end


to handle exitFullpage
     system s_page_flag, s_norm_b, s_norm_sz, s_old_f, currentrect
    set syslockscreen to true
    set s_page_flag to null
    hide recordfield title
   set s_old_f to 1
   set size of this book to defsize of this book
    set currentrect to "Inset Rectangle"
   send idle
    show rectangle "Inset Rectangle" of this background
    hide rectangle "Full Page" of this background
    hide scrollbar
end

to handle exitFullScreen
     system s_full_flag, s_norm_b, s_norm_sz, s_fullScreen, s_old_f
```

```
    set syslockscreen to true
    show recordfield "title" of this background
    set s_full_flag to null
    set s_page_flag to 1
    set s_old_f to 0
    --if s_norm_b is null
       --send sizeToPage
    --else
        --set bounds of mainwindow to s_norm_b
    --end
    hide rectangle "Full Rect" of this background
    hide scrollbar
end

to handle idle
     system s_old_is, s_old_vl, s_old_al, s_video_level, s_audio_level
     system s_video_zoomed, s_old_z, s_full_flag, s_old_f
     system s_page_flag, s_fullScreen, pom
      system s_suspend_fr, s_suspend_ch, s_videoDisc_init, currentrect
     if s_videoDisc_init is null
        if text of field "frame" of this background <> null
            --set text of field "frame" of this background to null
      end
   else
        if s_suspend_fr is null
           get videodisc("status frame")
          if it is "No response"
               request "Videodisc player off or disconnected."\
                && "Closing driver.  Videodisc controls will have no effect."
             get videodisc("Close")
            break idle
        end
           if isNumber(it) or it is null
               --set text of field "frame" of this background to it
        end
      end
   end
       get ResizeVideo(screenFromPage(sysWindowHandle,                \
                                sysPageScroll,              \
                                sysMagnification,           \
                                     bounds of rectangle currentrect of this background))
end

to handle sidle
     system s_old_is, s_old_vl, s_old_al, s_video_level, s_audio_level
     system s_video_zoomed, s_old_z, s_full_flag, s_old_f
     system s_page_flag, s_fullScreen, pom
      system s_suspend_fr, s_suspend_ch, s_videoDisc_init
     if s_videoDisc_init is null
        if text of field "frame" of this background <> null
            --set text of field "frame" of this background to null
```

```
    end
  else
      if s_suspend_fr is null
         get videodisc("status frame")
        if it is "No response"
             request "Videodisc player off or disconnected."\
              && "Closing driver.  Videodisc controls will have no effect."
           get videodisc("Close")
           break sidle
      end
        if isNumber(it) or it is null
            --set text of field "frame" of this background to it
      end
    end
  end
   conditions
   when s_old_f is 1
       get bounds of rectangle "Inset Rectangle" of this background
      get video("set inset", it)
      break conditions
   when s_page_flag is 1 -- <> null
       get bounds of rectangle "Full Page" of this background
      get video("set inset", it)
      break conditions
   when s_full_flag is 1 -- <> null
       get bounds of rectangle "Full Rect" of this background
      get video("set inset", it)
      break conditions
   end
   forward
end
```

```
--This script handles the behavior of the RECORD button for the German
--Video book
--It records and saves a sound clip in the target language for an instructor
--to use in a lesson segment.  This record and save option is only available
--when the book is in teacher/author mode, with the EDIT button active.  A
--student will be able to play back the sound clip associated with the page
--with which the instructor associated it.
--A student can use the RECORD button to record his/her own version of
--the target language recording; however, this sound clip is only available
--during one learning session.


to handle buttondown
    system s_wavFile, s_wavFileLen, s_wavRecording, editmode, s_wavPath
        get tbkWaveAudio("stop" && s_wavFile,"")
    if s_wavrecording is true then
        set s_wavrecording to false
        hide group recording of this background
            if editmode is native then
                get tbkWaveAudio("save" && s_wavFile,"")
            end
    else
            get tbkWaveAudio("close" && s_wavFile,"")
    set syslockscreen to true
            if editmode is student then
            set s_wavFile to "c:\lessons\german\student.wav"
            else
                set s_WavFile to s_wavPath & lessoncode of this book \
                & pagenumber of this page & ".wav"

            end
            ge  tbkWaveAudio("close" && s_wavFile,"")
    get tbkW veAudio("open" && s_wavFile && "Type waveaudio","")
    get tbkWaveAudio("set" && s_wavFile && "milliseconds","")
    get tbkWaveAudio("set" && s_wavFile && "samplespersec 22050","")
    get tbkWaveAudio("status" && s_wavFile && "samplespersec","")
            get tbkWaveAudio("set" && s_wavFile && "any input","")
            get tbkWaveAudio("cue" && s_wavFile && "input wait","")
            get tbkWaveAudio("record" && s_wavFile,"")
            set s_wavrecording to true
            show group "recording" of this background
            set a to sysmouseposition
    end
end


--Plans for a new feature include saving student files in an array of files
--for instructor review and evaluation:
--send record segment, endposition
--This records each student response starting after the previous and logs
```

```
--it into an array.  Playback then plays each segment from its
--startposition (endposition of previous segment) to its own endposition at
--the end the entire student response file is saved with its array for
--teacher evaluation.
--The native speaker inputs can be played back with the same type
--of array that the student has.
```

REFERENCES

[Af90]. United States Air Force Academy Department of Foreign Languages, GLM^2 Project Overview, 24 October 1990.

[Ba91]. Badler, Norman and Webber, Bonnie, "Task Communication Through Natural Language and Graphics," *AI Magazine* [Special issue], v. 11, no. 5, January 1991.

[Bg85]. Colorado University at Boulder Institute of Cognitive Science, *A Multimedia Knowledge Representation for an 'Intelligent' Computerized Tutor*, by P. Baggett and A. Ehrenfeucht, April 22, 1985.

[Bi90]. Bielawski, Larry and Lewand, Robert, *Intelligent Systems Design*, New York: John Wiley and Sons, 1990.

[Bm91]. "Basic Military Language Course Project Overview and Outline for Phased Courseware Development," projected development outline, rationale, and schedule for Special Forces Computer Based Training, Defense Language Institute, Presidio of Monterey, July 1991.

[Bo91]. Bonner, Fred, "IVD Learning: Design for Discovery," *Instruction Delivery Systems*, pp. 14-17,v. 5, no. 4, July/August 1991.

[Br82]. Barr, A., and Feigenbaum, E. A., *The Handbook of Artificial Intelligence*, v. 3, New York: William Kaufmann, Inc., 1982.

[Bu91]. Burns, Hugh and Parlett, James W., "The Evolution of Intelligent Tutoring Systems: Dimensions of Design." In *Intelligent Tutoring Systems: Evolutions In Design*, (pp. 1-12). Hillsdale, New Jersey: Lawrence Erlbaum Associates, 1991.

[By91]. Boy, Guy A., *Intelligent Assistant Systems*, Academic Press, 1991.

[Ca91]. Casner, Stephen., "A Task-Analytic approach to the Automated Design of Graphic Presentations," *ACM Transactions on Graphics*, pp. 111-151, v. 10, no. 2.

[Ck91]. Clark, David R., "The Demise of Multimedia," *IEEE Computer Graphics and Applications*, v. 11, no. 4, pp. 75-80, July 1991.

[Cn91]. Crane, Gregory, "Hypermedia and the Study of Ancient Culture," *IEEE Computer Graphics and Applications*, v. 11, no. 4, pp. 45-51, July 1991.

[Cr89]. Science Applications International Corporation Technical Report, *Applications of Artificial Intelligence to Foreign Language Learning: Analysis, Design, and Initial Prototyping*, by E. Criswell, A. Weinberg, L. Miller, H. Byrnes, and D. Blascak, 28 December1989.

[Cr, in press]. Criswell, Eleanor, Byrnes, Heidi, Pfister, Guenter (in press), "Intelligent Automated Strategies of Teaching Foreign Language in Context." In *Intelligent Tutoring Systems for Foreign Language Learning*. Berlin: Springer-Verlag.

[Dk91]. Duke University Computer Assisted Language Learning Project (DUCALL), Computer Assisted Language Instruction System (CALIS) version 2.23 User's Guide, June 1991.

[Er91]. Erickson, D. J., "Multimedia: The Next Wave," *Windows and OS/2 Magazine*, v. 2 no. 1, pp. 40-46, February/March 1991.

[Fa90]. Faris, Phillip. "Prototype, Arabic Language Instruction in Hypercard," CALL program under development at the Education Technology Division, Defense Language Institute, Presidio of Monterey.

[Fi90]. Fischer, Gerhard, "Communication Requirements for Cooperative Problem Solving Systems," *Information Systems*, v. 15, no. 1, 1990.

[Ga91]. Gayeski, Diane M., "Automating Instructional Design: A Case Study," *Instruction Delivery Systems*, pp. 20-24,v. 5, no. 4, July/August 1991.

[Ge87]. Gery, Gloria, *Making CBT Happen*. Boston: Weingarten Publications, 1987.

[Ha88]. Halff, Henry M.,"Curriculum and Instruction in Automated Tutors." In M.C. Polson & J.J. Richardson (Eds), *Foundations of Intelligent Tutoring Systems* (pp. 79-108). Hillsdale, New Jersey: Lawrence Erlbaum Associates, 1988.

[In89]. Inui, Masahiro, Miyasaka, Nobuji, Kawamara, Kasuhiko, & Bourne, John R. "Development of a Model-based Training System.," *Future Generation of Computer Systems*, v. 5, pp. 59-69, 1989.

[It90]. International Interactive Communications Society, *Interact Journal*, v. 2, Winter, 1989-90.

[Ka90]. Kahn, Paul. "Publishing Webs of Information at Brown University," *CD-ROM Professional*, pp. 80-86, September 1990.

[Ma91]. Marchionini, Gary, "Evaluating the Perseus Project," talk presented at Interactive MultiMedia '91 Conference, Arlington, Virginia, August 22, 1991.

[Me86]. Meyrowitz, Norman, "Intermedia: The Architecture and Construction of an Object-Oriented Hypermedia System and Applications Framework," *Object-Oriented Programming, Systems, an Applications Proceedings*, 1986.

[Mi88]. Micallef, J., "Encapsulation, Reuseability and Extensibility in Object-Oriented Programming Languages," Journal of Object-Oriented Programming, v. 1, no. 1, April/May 1988, pp. 12-35.

[Mr91]. Merrill, M. David. "Constructivism and Instructional Design," *Educational Technology*, pp. 45-52, May 1991.

[Ms91]. *Microsoft Multimedia Development Kit Programmer's Reference Version 1.0*, Microsoft Corporation, Redmon, Washington, 1991.

[Nc91]. National Cryptologic School, National Security Agency, *Project Replicate: Training Delivery System*, January 1991.

[Ne90]. Naval Postgraduate School Technical Report No. NPS52-90-024, *An Introduction to Object-Oriented Programming*, by Michel Nelson, April 1990.

[Ni90]. Nielsen, Jakob, *Hypertext and Hypermedia*, Academic Press, 1990.

[Ns91]. NSACSS Ft George G. Meade, MD Naval Message, Subject: Ensuring the Interchangeability of CBT Courseware Throughout the Cryptologic Training System, 171527Z Apr 91.

[Pe91]. Pea, Roy D.,"Learning Through Multimedia," *IEEE Computer Graphics and Applications*, v. 11, no. 4, pp. 58-66, July 1991.

[Ph91]. Phillips, Richard L., "MediaView, A General Multimedia Digital Publication System," *Communications of the ACM*, v. 34, no. 7, July 1991.

[Ps88]. Psotka, Joseph, Massey, L. Don, and Mutter, Sharon, (Eds.), *Intelligent Tutoring Systems: Lessons Learned*, Hillsdale, New Jersey: Lawrence Erlbaum Associates, 1988.

[Ps91]. Psotka, Joseph, "Natural Language Processing, Hypertext, and Language Instruction," *Journal of Interactive Instruction Development*, v. 1, no. 4, Spring 1989.

[Ra90]. Ragsdale, Daniel J., and Tidd, John P., *Designing Intelligent Computer Aided Instruction Systems with Integrated Knowledge Representation Schemes*, Master's Thesis, Naval Postgraduate School, Monterey, California, June 1990.

[Ru91]. Ruley, John D., "Windows Comes to Multimedia," *Windows and OS/2 Magazine*, v. 2 no. 1, pp. 60-64, February/March 1991.

[Sc91]. Schank, Roger C. and Jona, Menachem Y., "Empowering the Student: New Perspectives on the Design of Teaching Systems," *The Journal of the Learning Sciences*, v.1, no. 1, 1991.

[Sd91]. Scardamalia, Marlene and Bereiter, Carl, "Higher Levels of Agency for Children in Knowledge Buliding: A Challenge for the Design of New Knowledge Media," *The Journal of the Learning Sciences*, v.1, no. 1, 1991.

[Sg84]. Association for Computing Machinery Special Interest Group on Computer Graphics (ACM SIGGRAPH), *Computer Supported Design Exhibition Papers, ACM SIGGRAPH 84*, the Eleventh Annual Conference on Computer Graphics and Interactive Techniques, Minneapolis, Minnesota, 1984.

[Sl82]. Sleeman, D. and Brown, J.S. *Intelligent Tutoring Systems*. New York: Academic Press.

[St91]. Stephenson, Bruce W., "Multimedia at School, Will Our Kids Benefit?" *Windows and OS/2 Magazine*, v. 2 no. 1, pp. 72-74, February/March 1991.

[Sw91]. Swigger, Kathleen, "Managing Communcation Knowledge." In *Intelligent Tutoring Systems: Evolutions In Design*, (pp. 13-34). Hillsdale, New Jersey: Lawrence Erlbaum Associates, 1991.

[Wf91]. Woolf, Beverly Park, Soloway, Elliot, Clancey, William, Van Lehn, Kurt, and Suthers, Dan, "Knowledge-Based Environments for Teaching and Learning," AI Magazine [Special issue], v. 11, no. 5, January 1991.

[Wi90]. *CommandPost 7.0 User's Manual*, Wilson WindowWare, 14 May 1990.

[Wo91]. Woolsey, Kristina Hooper, "Multimedia Scouting," *IEEE Computer Graphics and Applications*, v. 11, no. 4, pp. 26-38, July 1991.

INITIAL DISTRIBUTION LIST

	No. Copies
1. Defense Technical Information Center Cameron Station Alexandria, Virginia 22314-6145	2
2. Library, Code 52 Naval Postgraduate School Monterey, California 93943-5100	2
3. Officer in Charge, Naval Security Group Detachment Monterey Presidio of Monterey, California 93944-5005	2
4. Yuh-jeng Lee Computer Science Department Code CSLe Naval Postgraduate School Monterey, California 93943-5000	8
5. David R. Pratt Computer Science Department Code CSPr Naval Postgraduate School Monterey, California 93943-5000	1